

```

var unityFramework = (() => {
  var _scriptDir = typeof document !== 'undefined' && document.currentScript ?
    document.currentScript.src : undefined;
  if (typeof __filename !== 'undefined') _scriptDir = _scriptDir || __filename;
  return (
    function(unityFramework) {
      unityFramework = unityFramework || {};
    }
  );
}

// The Module object: Our interface to the outside world. We import
// and export values on it. There are various ways Module can be used:
// 1. Not defined. We create it here
// 2. A function parameter, function(Module) { ..generated code.. }
// 3. pre-run appended it, var Module = {}; ..generated code..
// 4. External script tag defines var Module.
// We need to check if Module already exists (e.g. case 3 above).
// Substitution will be replaced with actual code on later stage of the build,
// this way Closure Compiler will not mangle it (e.g. case 4. above).
// Note that if you want to run closure, and also to use Module
// after the generated code, you will need to define var Module = {};
// before the code. Then that object will be used in the code, and you
// can continue to use Module afterwards as well.
var Module = typeof unityFramework !== 'undefined' ? unityFramework : {};

// See https://caniuse.com/mdn-javascript_builtins_object_assign

// Set up the promise that indicates the Module is initialized
var readyPromiseResolve, readyPromiseReject;
Module['ready'] = new Promise(function(resolve, reject) {
  readyPromiseResolve = resolve;
  readyPromiseReject = reject;
});

  if (!Object.getOwnPropertyDescriptor(Module['ready'], '_main')) {
    Object.defineProperty(Module['ready'], '_main', { configurable: true, get:
      function() { abort('You are getting _main on the Promise object, instead of the
        instance. Use .then() to get called back with the instance, see the MODULARIZE docs in
        src/settings.js') } });
    Object.defineProperty(Module['ready'], '_main', { configurable: true, set:
      function() { abort('You are setting _main on the Promise object, instead of the
        instance. Use .then() to get called back with the instance, see the MODULARIZE docs in
        src/settings.js') } });
  }

  if (!Object.getOwnPropertyDescriptor(Module['ready'], '_getMemInfo')) {
    Object.defineProperty(Module['ready'], '_getMemInfo', { configurable: true,
      get: function() { abort('You are getting _getMemInfo on the Promise object, instead of
        the instance. Use .then() to get called back with the instance, see the MODULARIZE docs in
        src/settings.js') } });
  }
}

```

```

Object.defineProperty(Module['ready'], '_getMemInfo', { configurable: true,
set: function() { abort('You are setting _getMemInfo on the Promise object, instead of
the instance. Use .then() to get called back with the instance, see the MODULARIZE docs
in src/settings.js') } });
}

if (!Object.getOwnPropertyDescriptor(Module['ready'], '_SendMessageFloat')) {
    Object.defineProperty(Module['ready'], '_SendMessageFloat', { configurable:
true, get: function() { abort('You are getting _SendMessageFloat on the Promise object,
instead of the instance. Use .then() to get called back with the instance, see the
MODULARIZE docs in src/settings.js') } });
    Object.defineProperty(Module['ready'], '_SendMessageFloat', { configurable:
true, set: function() { abort('You are setting _SendMessageFloat on the Promise object,
instead of the instance. Use .then() to get called back with the instance, see the
MODULARIZE docs in src/settings.js') } });
}

if (!Object.getOwnPropertyDescriptor(Module['ready'], '_SendMessageString')) {
    Object.defineProperty(Module['ready'], '_SendMessageString', { configurable:
true, get: function() { abort('You are getting _SendMessageString on the Promise
object, instead of the instance. Use .then() to get called back with the instance, see
the MODULARIZE docs in src/settings.js') } });
    Object.defineProperty(Module['ready'], '_SendMessageString', { configurable:
true, set: function() { abort('You are setting _SendMessageString on the Promise
object, instead of the instance. Use .then() to get called back with the instance, see
the MODULARIZE docs in src/settings.js') } });
}

if (!Object.getOwnPropertyDescriptor(Module['ready'], '_SendMessage')) {
    Object.defineProperty(Module['ready'], '_SendMessage', { configurable: true,
get: function() { abort('You are getting _SendMessage on the Promise object, instead of
the instance. Use .then() to get called back with the instance, see the MODULARIZE docs
in src/settings.js') } });
    Object.defineProperty(Module['ready'], '_SendMessage', { configurable: true,
set: function() { abort('You are setting _SendMessage on the Promise object, instead of
the instance. Use .then() to get called back with the instance, see the MODULARIZE docs
in src/settings.js') } });
}

if (!Object.getOwnPropertyDescriptor(Module['ready'], '_SetFullscreen')) {
    Object.defineProperty(Module['ready'], '_SetFullscreen', { configurable: true,
get: function() { abort('You are getting _SetFullscreen on the Promise object, instead
of the instance. Use .then() to get called back with the instance, see the MODULARIZE
docs in src/settings.js') } });
    Object.defineProperty(Module['ready'], '_SetFullscreen', { configurable: true,
set: function() { abort('You are setting _SetFullscreen on the Promise object, instead
of the instance. Use .then() to get called back with the instance, see the MODULARIZE
docs in src/settings.js') } });
}

if (!Object.getOwnPropertyDescriptor(Module['ready'], '_InjectProfilerSample')) {

```

```

Object.defineProperty(Module['ready'], '_InjectProfilerSample', { configurable: true, get: function() { abort('You are getting _InjectProfilerSample on the Promise object, instead of the instance. Use .then() to get called back with the instance, see the MODULARIZE docs in src/settings.js') } });
Object.defineProperty(Module['ready'], '_InjectProfilerSample', { configurable: true, set: function() { abort('You are setting _InjectProfilerSample on the Promise object, instead of the instance. Use .then() to get called back with the instance, see the MODULARIZE docs in src/settings.js') } });
}

if (!Object.getOwnPropertyDescriptor(Module['ready'], '__stdio_exit')) {
    Object.defineProperty(Module['ready'], '__stdio_exit', { configurable: true, get: function() { abort('You are getting __stdio_exit on the Promise object, instead of the instance. Use .then() to get called back with the instance, see the MODULARIZE docs in src/settings.js') } });
    Object.defineProperty(Module['ready'], '__stdio_exit', { configurable: true, set: function() { abort('You are setting __stdio_exit on the Promise object, instead of the instance. Use .then() to get called back with the instance, see the MODULARIZE docs in src/settings.js') } });
}

if (!Object.getOwnPropertyDescriptor(Module['ready'], 'onRuntimeInitialized')) {
    Object.defineProperty(Module['ready'], 'onRuntimeInitialized', { configurable: true, get: function() { abort('You are getting onRuntimeInitialized on the Promise object, instead of the instance. Use .then() to get called back with the instance, see the MODULARIZE docs in src/settings.js') } });
    Object.defineProperty(Module['ready'], 'onRuntimeInitialized', { configurable: true, set: function() { abort('You are setting onRuntimeInitialized on the Promise object, instead of the instance. Use .then() to get called back with the instance, see the MODULARIZE docs in src/settings.js') } });
}

// --pre-jses are emitted after the Module integration code, so that they can
// refer to Module (if they choose; they can also define Module)

// Emscripten 1.x had a function Pointer_stringify() to marshal C strings to JS
// strings. That has been obsoleted by the new UTF8/16/32ToString() API family.
function Pointer_stringify(s, len) {
    warnOnce("The JavaScript function 'Pointer_stringify(ptrToSomeCString)' is
    obsoleted and will be removed in a future Unity version. Please call
    'UTF8ToString(ptrToSomeCString)' instead.");
    return UTF8ToString(s, len);
}
Module['Pointer_stringify'] = Pointer_stringify;
var stackTraceReference =
"(^|\n)(\\s+\\|)jsStackTrace(\\s+\\(|@)([^\\n]+):\\d+:\\d+(\\))|(\n|$)";
var stackTraceReferenceMatch = jsStackTrace().match(new RegExp(stackTraceReference));
if (stackTraceReferenceMatch)
    Module.stackTraceRegExp = new RegExp(stackTraceReference.replace("[^\\n]+",
    stackTraceReferenceMatch[4].replace(/\^\$[\[][\]]/.g, "\\$&").replace("jsStackTrac",
    "[^\\n]+"));

```

```

var abort = function (what) {
    if (ABORT)
        return;
    ABORT = true;
    EXITSTATUS = 1;
    if (typeof ENVIRONMENT_IS_PTHREAD !== "undefined" && ENVIRONMENT_IS_PTHREAD)
        console.error("Pthread aborting at " + new Error().stack);
    if (what !== undefined) {
        out(what);
        err(what);
        what = JSON.stringify(what)
    } else {
        what = "";
    }
    var message = "abort(" + what + ") at " + stackTrace();
    if (Module.abortHandler && Module.abortHandler(message))
        return;
    throw message;
}

Module["SetFullscreen"] = function (fullscreen) {
    if (typeof runtimeInitialized === 'undefined' || !runtimeInitialized) {
        console.log ("Runtime not initialized yet.");
    } else if (typeof JSEvents === 'undefined') {
        console.log ("Player not loaded yet.");
    } else {
        var tmp = JSEvents.canPerformEventHandlerRequests;
        JSEvents.canPerformEventHandlerRequests = function () { return 1; };
        Module.ccall("SetFullscreen", null, ["number"], [fullscreen]);
        JSEvents.canPerformEventHandlerRequests = tmp;
    }
};

if (!Module['ENVIRONMENT_IS_PTHREAD']) {
    Module['preRun'].push(function () {
        // Initialize the IndexedDB based file system. Module['unityFileSystemInit'] allows
        // developers to override this with their own function, when they want to do cloud
        storage
        // instead.
        var unityFileSystemInit = Module['unityFileSystemInit'] || function () {
            FS.mkdir('/idbfs');
            FS.mount(IDBFS, {}, '/idbfs');
            Module.addRunDependency('JS_FileSystem_Mount');
            FS.syncfs(true, function (err) {
                if (err)
                    console.log('IndexedDB is not available. Data will not persist in cache and
PlayerPrefs will not be saved.');
                Module.removeRunDependency('JS_FileSystem_Mount');
            });
        };
        unityFileSystemInit();
    });
}

var videoInputDevices = []; // Set to null to disable video input devices altogether.
var videoInputDevicesEnumerated = false;

```

```

var removeEnumerateMediaDevicesRunDependency;
var enumerateWatchdog = null;

// Bug/limitation: Chrome does not specify deviceIds for any MediaDeviceInfo input
// devices at least in Chrome 85 on Windows 10
// This means that we need to use an awkward heuristic way of matching old video input
// connections to new ones.
function matchToOldDevice(newDevice) {
    var oldDevices = Object.keys(videoInputDevices);
    // First match by deviceId
    for(var i = 0; i < oldDevices.length; ++i) {
        var old = videoInputDevices[oldDevices[i]];
        if (old.deviceId && old.deviceId == newDevice.deviceId) return old;
    }
    // Then by object identity, in case that is supported.
    for(var i = 0; i < oldDevices.length; ++i) {
        var old = videoInputDevices[oldDevices[i]];
        if (old == newDevice) return old;
    }
    // Then by label
    for(var i = 0; i < oldDevices.length; ++i) {
        var old = videoInputDevices[oldDevices[i]];
        if (old.label && old.label == newDevice.label) return old;
    }
    // Last, by groupId + kind combination
    for(var i = 0; i < oldDevices.length; ++i) {
        var old = videoInputDevices[oldDevices[i]];
        if (old.groupId && old.kind && old.groupId == newDevice.groupId && old.kind
== newDevice.kind) return old;
    }
}

function assignNewVideoInputId() {
    for(var i = 0;; ++i) {
        if (!videoInputDevices[i]) return i;
    }
}

function updateVideoInputDevices(devices) {
    removeEnumerateMediaDevicesRunDependency();
    // we're going to clear the list of videoInputDevices and regenerate it to get
    more accurate info after being granted camera access
    videoInputDevices = [];
    var retainedDevices = {};
    var newDevices = [];

    // Find devices that still exist
    devices.forEach(function (device) {
        if (device.kind === 'videoinput') { // Only interested in WebCam inputs
            var oldDevice = matchToOldDevice(device);
            if (oldDevice) {
                retainedDevices[oldDevice.id] = oldDevice;
            } else {
                newDevices.push(device);
            }
        }
    })
}

```

```

        }
    });
    videoInputDevices = retainedDevices;

    // Assign IDs to video input devices that are new
    newDevices.forEach(function (device) {
        if (!device.id) {
            device.id = assignNewVideoInputId();
            // Attempt to name the device. In both Firefox and Chrome, label is
            null.
            // In Chrome, deviceId is null. (could use it here, but human-
            readable
            // name is probably better than a long hash)
            device.name = device.label || ("Video input #" + (device.id + 1));

            // Chrome 85 on Android labels camera provides devices with labels
            like
            // "camera2 0, facing back" and "camera2 1, facing front", use that
            to
            // determine whether the device is front facing or not.
            // some labels don't provide that info, like the camera on a 2019
            MacbookPro: FaceTime HD Camera (Built-in)
            // so if there's no "front" or "back" in the label or name, we're
            going to assume it's front facing

            device.isFrontFacing = device.name.toLowerCase().includes('front') ||
            !(device.name.toLowerCase().includes('front')) &&
            !(device.name.toLowerCase().includes('back')));

            videoInputDevices[device.id] = device;
        }
    });
}

function enumerateMediaDeviceList() {
    if (!videoInputDevices) return;
    // Bug/limitation: If there are multiple video or audio devices connected,
    // Chrome only lists one of each category (audioinput/videoinput/audioutput)
    (tested Chrome 85 on Windows 10)
    navigator.mediaDevices.enumerateDevices().then(function(devices) {
        updateVideoInputDevices(devices);
        videoInputDevicesEnumerated = true;
    }).catch(function(e) {
        console.warn('Unable to enumerate media devices: ' + e + '\nWebcams will
not be available.');
        disableAccessToMediaDevices();
    });

    // Work around Firefox 81 bug on Windows:
    // https://bugzilla.mozilla.org/show_bug.cgi?id=1397977, devicechange
    // events do not fire, so resort to polling for device changes once every
    // 60 seconds.
    if (/Firefox/.test(navigator.userAgent)) {
        setTimeout(enumerateMediaDeviceList, 60000);
        warnOnce('Applying workaround to Firefox bug
https://bugzilla.mozilla.org/show_bug.cgi?id=1397977');
    }
}

```



```

function disableAccessToMediaDevices() {
    // Safari 11 has navigator.mediaDevices, but
    // navigator.mediaDevices.add/removeEventListener is undefined
    if (navigator.mediaDevices && navigator.mediaDevices.removeEventListener) {
        navigator.mediaDevices.removeEventListener('devicechange',
        enumerateMediaDeviceList);
    }
    videoInputDevices = null;
}
Module['disableAccessToMediaDevices'] = disableAccessToMediaDevices;

if (!navigator.mediaDevices) {
    console.warn('navigator.mediaDevices not supported by this browser. Webcam access
will not be available.' + (location.protocol == 'https:' ? '' : ' Try hosting the page
over HTTPS, because some browsers disable webcam access when insecure HTTP is being
used.'));
    disableAccessToMediaDevices();
} else if (typeof ENVIRONMENT_IS_PTHREAD === "undefined" || !ENVIRONMENT_IS_PTHREAD)
setTimeout(function() {
    try {
        // Do not start engine main() until we have completed enumeration.
        addRunDependency('enumerateMediaDevices');
        removeEnumerateMediaDevicesRunDependency = function() {
            if (enumerateWatchdog !== null) clearTimeout(enumerateWatchdog);
            removeRunDependency('enumerateMediaDevices');
            if (navigator.mediaDevices) console.log('navigator.mediaDevices
support available');
            removeEnumerateMediaDevicesRunDependency = function() {};
        }
        // Enumerate media devices now at startup..
        enumerateMediaDeviceList();
        // Firefox won't complete device enumeration if the window isn't in focus
        causing the startup to hang, so we
        // wait a second before removing the dependency and starting with an empty
        list of devices. Moving forward it's
        // likely more browsers will assume this standard.
        // See https://w3c.github.io/mediacapture-main/#dom-mediadevices-
enumerateddevices
        enumerateWatchdog = setTimeout(removeEnumerateMediaDevicesRunDependency,
        1000);

        // .. and whenever the connected devices list changes.
        navigator.mediaDevices.addEventListener('devicechange',
        enumerateMediaDeviceList);
    } catch(e) {
        console.warn('Unable to enumerate media devices: ' + e);
        disableAccessToMediaDevices();
    }
}, 0);

function SendMessage(gameObject, func, param)
{
    if (param === undefined)
        Module.ccall("SendMessage", null, ["string", "string"], [gameObject, func]);
    else if (typeof param === "string")
        Module.ccall("SendMessageString", null, ["string", "string", "string"],
[gameObject, func, param]);
    else if (typeof param === "number")
        Module.ccall("SendMessageFloat", null, ["string", "string", "number"],

```

```
[gameObject, func, param]);
```

```

    else
        throw "" + param + " is does not have a type which is supported by
SendMessageBox.";
}
Module["SendMessageBox"] = SendMessageBox; // to avoid emscripten stripping

// Sometimes an existing Module object exists with properties
// meant to overwrite the default module functionality. Here
// we collect those properties and reapply _after_ we configure
// the current environment's defaults to avoid having to be so
// defensive during initialization.
var moduleOverrides = Object.assign({}, Module);

var arguments_ = [];
var thisProgram = './this.program';
var quit_ = (status, toThrow) => {
    throw toThrow;
};

// Determine the runtime environment we are in. You can customize this by
// setting the ENVIRONMENT setting at compile time (see settings.js).

// Attempt to auto-detect the environment
var ENVIRONMENT_IS_WEB = typeof window == 'object';
var ENVIRONMENT_IS_WORKER = typeof importScripts == 'function';
// N.b. Electron.js environment is simultaneously a NODE-environment, but
// also a web environment.
var ENVIRONMENT_IS_NODE = typeof process == 'object' && typeof process.versions ==
'object' && typeof process.versions.node == 'string';
var ENVIRONMENT_IS_SHELL = !ENVIRONMENT_IS_WEB && !ENVIRONMENT_IS_NODE &&
!ENVIRONMENT_IS_WORKER;

if (Module['ENVIRONMENT']) {
    throw new Error('Module.ENVIRONMENT has been deprecated. To force the environment,
use the ENVIRONMENT compile-time option (for example, -s ENVIRONMENT=web or -s
ENVIRONMENT=node)');
}

// `/` should be present at the end if `scriptDirectory` is not empty
var scriptDirectory = '';
function locateFile(path) {
    if (Module['locateFile']) {
        return Module['locateFile'](path, scriptDirectory);
    }
    return scriptDirectory + path;
}

// Hooks that are implemented differently in different runtime environments.
var read_,
    readAsync,
    readBinary,
    setWindowTitle;

// Normally we don't log exceptions but instead let them bubble out the top
// level where the embedding environment (e.g. the browser) can handle

```

```

// them.
// However under v8 and node we sometimes exit the process directly in which case
// its up to use us to log the exception before exiting.
// If we fix https://github.com/emscripten-core/emscripten/issues/15080
// this may no longer be needed under node.
function logExceptionOnExit(e) {
    if (e instanceof ExitStatus) return;
    let toLog = e;
    if (e && typeof e == 'object' && e.stack) {
        toLog = [e, e.stack];
    }
    err('exiting due to exception: ' + toLog);
}

var fs;
var nodePath;
var requireNodeFS;

if (ENVIRONMENT_IS_NODE) {
    if (!(typeof process == 'object' && typeof require == 'function')) throw new
Error('not compiled for this environment (did you build to HTML and try to run it not
on the web, or set ENVIRONMENT to something - like node - and run it someplace else -
like on the web?)');
    if (ENVIRONMENT_IS_WORKER) {
        scriptDirectory = require('path').dirname(scriptDirectory) + '/';
    } else {
        scriptDirectory = __dirname + '/';
    }
}

// include: node_shell_read.js

requireNodeFS = () => {
    // Use nodePath as the indicator for these not being initialized,
    // since in some environments a global fs may have already been
    // created.
    if (!nodePath) {
        fs = require('fs');
        nodePath = require('path');
    }
};

read_ = function shell_read(filename, binary) {
    requireNodeFS();
    filename = nodePath['normalize'](filename);
    return fs.readFileSync(filename, binary ? undefined : 'utf8');
};

readBinary = (filename) => {
    var ret = read_(filename, true);
    if (!ret.buffer) {
        ret = new Uint8Array(ret);
    }
    assert(ret.buffer);
    return ret;
}

```

```

};

readAsync = (filename, onload, onerror) => {
    requireNodeFS();
    filename = nodePath['normalize'](filename);
    fs.readFile(filename, function(err, data) {
        if (err) onerror(err);
        else onload(data.buffer);
    });
};

// end include: node_shell_read.js
if (process['argv'].length > 1) {
    thisProgram = process['argv'][1].replace(/\//g, '/');
}

arguments_ = process['argv'].slice(2);

// MODULARIZE will export the module in the proper place outside, we don't need to
// export here

process['on']('uncaughtException', function(ex) {
    // suppress ExitStatus exceptions from showing an error
    if (!(ex instanceof ExitStatus)) {
        throw ex;
    }
});

// Without this older versions of node (< v15) will log unhandled rejections
// but return 0, which is not normally the desired behaviour. This is
// not be needed with node v15 and about because it is now the default
// behaviour:
// See https://nodejs.org/api/cli.html#cliUnhandledRejectionsMode
process['on']('unhandledRejection', function(reason) { throw reason; });

quit_ = (status, toThrow) => {
    if (keepRuntimeAlive()) {
        process['exitCode'] = status;
        throw toThrow;
    }
    logExceptionOnExit(toThrow);
    process['exit'](status);
};

Module['inspect'] = function () { return '[Emscripten Module object]'; };

} else
if (ENVIRONMENT_IS_SHELL) {

    if ((typeof process == 'object' && typeof require === 'function') || typeof window ==
    'object' || typeof importScripts === 'function') throw new Error('not compiled for this
environment (did you build to HTML and try to run it not on the web, or set ENVIRONMENT
to something - like node - and run it someplace else - like on the web?)');

    if (typeof read != 'undefined') {

```

```

read_ = function shell_read(f) {
    return read(f);
};

readBinary = function readBinary(f) {
    let data;
    if (typeof readbuffer == 'function') {
        return new Uint8Array(readbuffer(f));
    }
    data = read(f, 'binary');
    assert(typeof data == 'object');
    return data;
};

readAsync = function readAsync(f, onload, onerror) {
    setTimeout(() => onload(readBinary(f)), 0);
};

if (typeof scriptArgs != 'undefined') {
    arguments_ = scriptArgs;
} else if (typeof arguments != 'undefined') {
    arguments_ = arguments;
}

if (typeof quit == 'function') {
    quit_ = (status, toThrow) => {
        logExceptionOnExit(toThrow);
        quit(status);
    };
}

if (typeof print != 'undefined') {
    // Prefer to use print/printErr where they exist, as they usually work better.
    if (typeof console == 'undefined') console = /** @type{!Console} */({});

    console.log = /** @type{!function(this:Console, ...): undefined} */(print);
    console.warn = console.error = /** @type{!function(this:Console, ...): undefined} */
*/ (typeof printErr != 'undefined' ? printErr : print);
}

} else

// Note that this includes Node.js workers when relevant (pthreads is enabled).
// Node.js workers are detected as a combination of ENVIRONMENT_IS_WORKER and
// ENVIRONMENT_IS_NODE.
if (ENVIRONMENT_IS_WEB || ENVIRONMENT_IS_WORKER) {
    if (ENVIRONMENT_IS_WORKER) { // Check worker, not web, since window could be
polyfilled
        scriptDirectory = self.location.href;
    } else if (typeof document != 'undefined' && document.currentScript) { // web
        scriptDirectory = document.currentScript.src;
    }
    // When MODULARIZE, this JS may be executed later, after document.currentScript
    // is gone, so we saved it, and we use it here instead of any other info.
    if (_scriptDir) {

```

```

        scriptDirectory = _scriptDir;
    }
    // blob urls look like blob:http://site.com/etc/etc and we cannot infer anything from
    // them.
    // otherwise, slice off the final part of the url to find the script directory.
    // if scriptDirectory does not contain a slash, lastIndexOf will return -1,
    // and scriptDirectory will correctly be replaced with an empty string.
    // If scriptDirectory contains a query (starting with ?) or a fragment (starting with
    #),
    // they are removed because they could contain a slash.
    if (scriptDirectory.indexOf('blob:') !== 0) {
        scriptDirectory = scriptDirectory.substr(0, scriptDirectory.replace(/[^#].*/,
        "")).lastIndexOf('/')+1);
    } else {
        scriptDirectory = '';
    }

    if (!(typeof window == 'object' || typeof importScripts == 'function')) throw new
Error('not compiled for this environment (did you build to HTML and try to run it not
on the web, or set ENVIRONMENT to something - like node - and run it someplace else -
like on the web?)');

    // Differentiate the Web Worker from the Node Worker case, as reading must
    // be done differently.
    {
// include: web_or_worker_shell_read.js

read_ = (url) => {
    var xhr = new XMLHttpRequest();
    xhr.open('GET', url, false);
    xhr.send(null);
    return xhr.responseText;
}

if (ENVIRONMENT_IS_WORKER) {
    readBinary = (url) => {
        var xhr = new XMLHttpRequest();
        xhr.open('GET', url, false);
        xhr.responseType = 'arraybuffer';
        xhr.send(null);
        return new Uint8Array/** @type{!ArrayBuffer} */(xhr.response));
    };
}

readAsync = (url, onload, onerror) => {
    var xhr = new XMLHttpRequest();
    xhr.open('GET', url, true);
    xhr.responseType = 'arraybuffer';
    xhr.onload = () => {
        if (xhr.status == 200 || (xhr.status == 0 && xhr.response)) { // file URLs can
return 0
            onload(xhr.response);
            return;
        }
    }
}

```

```

        onerror();
    };
    xhr.onerror = onerror;
    xhr.send(null);
}

// end include: web_or_worker_shell_read.js
}

setWindowTitle = (title) => document.title = title;
} else
{
    throw new Error('environment detection error');
}

var out = Module['print'] || console.log.bind(console);
var err = Module['printErr'] || console.warn.bind(console);

// Merge back in the overrides
Object.assign(Module, moduleOverrides);
// Free the object hierarchy contained in the overrides, this lets the GC
// reclaim data used e.g. in memoryInitializerRequest, which is a large typed array.
moduleOverrides = null;
checkIncomingModuleAPI();

// Emit code to handle expected values on the Module object. This applies Module.x
// to the proper local x. This has two benefits: first, we only emit it if it is
// expected to arrive, and second, by using a local everywhere else that can be
// minified.

if (Module['arguments']) arguments_ = Module['arguments']; legacyModuleProp('arguments',
'arguments_');

if (Module['thisProgram']) thisProgram =
Module['thisProgram']; legacyModuleProp('thisProgram', 'thisProgram');

if (Module['quit']) quit_ = Module['quit']; legacyModuleProp('quit', 'quit_');

// perform assertions in shell.js after we set up out() and err(), as otherwise if an
assertion fails it cannot print the message
// Assertions on removed incoming Module JS APIs.
assert(typeof Module['memoryInitializerPrefixURL'] == 'undefined',
'Module.memoryInitializerPrefixURL option was removed, use Module.locateFile instead');
assert(typeof Module['pthreadMainPrefixURL'] == 'undefined',
'Module(pthreadMainPrefixURL option was removed, use Module.locateFile instead');
assert(typeof Module['cdInitializerPrefixURL'] == 'undefined',
'Module.cdInitializerPrefixURL option was removed, use Module.locateFile instead');
assert(typeof Module['filePackagePrefixURL'] == 'undefined',
'Module.filePackagePrefixURL option was removed, use Module.locateFile instead');
assert(typeof Module['read'] == 'undefined', 'Module.read option was removed (modify
read_ in JS)');
assert(typeof Module['readAsync'] == 'undefined', 'Module.readAsync option was removed
(modify readAsync in JS)');
assert(typeof Module['readBinary'] == 'undefined', 'Module.readBinary option was
removed (modify readBinary in JS)');

```

```

assert(typeof Module['setWindowTitle'] == 'undefined', 'Module.setWindowTitle option  
was removed (modify setWindowTitle in JS)');  
assert(typeof Module['TOTAL_MEMORY'] == 'undefined', 'Module.TOTAL_MEMORY has been  
renamed Module.INITIAL_MEMORY');  
legacyModuleProp('read', 'read_');  
legacyModuleProp('readAsync', 'readAsync');  
legacyModuleProp('readBinary', 'readBinary');  
legacyModuleProp('setWindowTitle', 'setWindowTitle');

var PROXYFS = 'PROXYFS is no longer included by default; build with -lproxyfs.js';  
var WORKERFS = 'WORKERFS is no longer included by default; build with -lworkerfs.js';  
var NODEFS = 'NODEFS is no longer included by default; build with -lnodefs.js';

assert(!ENVIRONMENT_IS_SHELL, "shell environment detected but not enabled at build  
time. Add 'shell' to `--s ENVIRONMENT` to enable.");

var STACK_ALIGN = 16;  
var POINTER_SIZE = 4;

function getNativeTypeSize(type) {  
    switch (type) {  
        case 'i1': case 'i8': return 1;  
        case 'i16': return 2;  
        case 'i32': return 4;  
        case 'i64': return 8;  
        case 'float': return 4;  
        case 'double': return 8;  
        default: {  
            if (type[type.length - 1] === '*') {  
                return POINTER_SIZE;  
            } else if (type[0] === 'i') {  
                const bits = Number(type.substr(1));  
                assert(bits % 8 === 0, 'getNativeTypeSize invalid bits ' + bits + ', type ' +  
type);  
                return bits / 8;  
            } else {  
                return 0;  
            }  
        }  
    }  
}  
}

function warnOnce(text) {  
    if (!warnOnce.shown) warnOnce.shown = {};  
    if (!warnOnce.shown[text]) {  
        warnOnce.shown[text] = 1;  
        err(text);  
    }  
}

// include: runtime_functions.js

```

```

// Wraps a JS function as a wasm function with a given signature.
function convertJsFunctionToWasm(func, sig) {

    // If the type reflection proposal is available, use the new
    // "WebAssembly.Function" constructor.
    // Otherwise, construct a minimal wasm module importing the JS function and
    // re-exporting it.
    if (typeof WebAssembly.Function == "function") {
        var typeNames = {
            'i': 'i32',
            'j': 'i64',
            'f': 'f32',
            'd': 'f64'
        };
        var type = {
            parameters: [],
            results: sig[0] == 'v' ? [] : [typeNames[sig[0]]]
        };
        for (var i = 1; i < sig.length; ++i) {
            type.parameters.push(typeNames[sig[i]]);
        }
        return new WebAssembly.Function(type, func);
    }

    // The module is static, with the exception of the type section, which is
    // generated based on the signature passed in.
    var typeSection = [
        0x01, // id: section,
        0x00, // length: 0 (placeholder)
        0x01, // count: 1
        0x60, // form: func
    ];
    var sigRet = sig.slice(0, 1);
    var sigParam = sig.slice(1);
    var typeCodes = {
        'i': 0x7f, // i32
        'j': 0x7e, // i64
        'f': 0x7d, // f32
        'd': 0x7c, // f64
    };

    // Parameters, length + signatures
    typeSection.push(sigParam.length);
    for (var i = 0; i < sigParam.length; ++i) {
        typeSection.push(typeCodes[sigParam[i]]);
    }

    // Return values, length + signatures
    // With no multi-return in MVP, either 0 (void) or 1 (anything else)
    if (sigRet == 'v') {
        typeSection.push(0x00);
    } else {
        typeSection = typeSection.concat([0x01, typeCodes[sigRet]]);
    }
}

```

```

}

// Write the overall length of the type section back into the section header
// (excepting the 2 bytes for the section id and length)
typeSection[1] = typeSection.length - 2;

// Rest of the module is static
var bytes = new Uint8Array([
  0x00, 0x61, 0x73, 0x6d, // magic ("\0asm")
  0x01, 0x00, 0x00, 0x00, // version: 1
].concat(typeSection, [
  0x02, 0x07, // import section
  // (import "e" "f" (func 0 (type 0)))
  0x01, 0x01, 0x65, 0x01, 0x66, 0x00, 0x00,
  0x07, 0x05, // export section
  // (export "f" (func 0 (type 0)))
  0x01, 0x01, 0x66, 0x00, 0x00,
]));
);

// We can compile this wasm module synchronously because it is very small.
// This accepts an import (at "e.f"), that it reroutes to an export (at "f")
var module = new WebAssembly.Module(bytes);
var instance = new WebAssembly.Instance(module, {
  'e': {
    'f': func
  }
});
var wrappedFunc = instance.exports['f'];
return wrappedFunc;
}

var freeTableIndexes = [];

// Weak map of functions in the table to their indexes, created on first use.
var functionsInTableMap;

function getEmptyTableSlot() {
  // Reuse a free index if there is one, otherwise grow.
  if (freeTableIndexes.length) {
    return freeTableIndexes.pop();
  }
  // Grow the table
  try {
    wasmTable.grow(1);
  } catch (err) {
    if (!(err instanceof RangeError)) {
      throw err;
    }
    throw 'Unable to grow wasm table. Set ALLOW_TABLE_GROWTH.';
  }
  return wasmTable.length - 1;
}

function updateTableMap(offset, count) {
  for (var i = offset; i < offset + count; i++) {

```

```

    var item = getWasmTableEntry(i);
    // Ignore null values.
    if (item) {
        functionsInTableMap.set(item, i);
    }
}

/***
 * Add a function to the table.
 * 'sig' parameter is required if the function being added is a JS function.
 * @param {string=} sig
 */
function addFunction(func, sig) {
    assert(typeof func != 'undefined');

    // Check if the function is already in the table, to ensure each function
    // gets a unique index. First, create the map if this is the first use.
    if (!functionsInTableMap) {
        functionsInTableMap = new WeakMap();
        updateTableMap(0, wasmTable.length);
    }
    if (functionsInTableMap.has(func)) {
        return functionsInTableMap.get(func);
    }

    // It's not in the table, add it now.

    var ret = getEmptyTableSlot();

    // Set the new value.
    try {
        // Attempting to call this with JS function will cause of table.set() to fail
        setWasmTableEntry(ret, func);
    } catch (err) {
        if (!(err instanceof TypeError)) {
            throw err;
        }
        assert(typeof sig != 'undefined', 'Missing signature argument to addFunction: ' +
func);
        var wrapped = convertJsFunctionToWasm(func, sig);
        setWasmTableEntry(ret, wrapped);
    }

    functionsInTableMap.set(func, ret);

    return ret;
}

function removeFunction(index) {
    functionsInTableMap.delete(getWasmTableEntry(index));
    freeTableIndexes.push(index);
}

// end include: runtime_functions.js

```

```

// include: runtime_debug.js

function legacyModuleProp(prop, newName) {
  if (!Object.getOwnPropertyDescriptor(Module, prop)) {
    Object.defineProperty(Module, prop, {
      configurable: true,
      get: function() {
        abort('Module.' + prop + ' has been replaced with plain ' + newName + ' (the
initial value can be provided on Module, but after startup the value is only looked for
on a local variable of that name)');
      }
    });
  }
}

function ignoredModuleProp(prop) {
  if (Object.getOwnPropertyDescriptor(Module, prop)) {
    abort(`Module.${prop} was supplied but ${prop} not included in
INCOMING_MODULE_JS_API`);
  }
}

function unexportedMessage(sym, isFSSybol) {
  var msg = `!${sym} was not exported. add it to EXPORTED_RUNTIME_METHODS (see
the FAQ)`;
  if (isFSSybol) {
    msg += '. Alternatively, forcing filesystem support (-s FORCE_FILESYSTEM=1) can
export this for you';
  }
  return msg;
}

function unexportedRuntimeSymbol(sym, isFSSybol) {
  if (!Object.getOwnPropertyDescriptor(Module, sym)) {
    Object.defineProperty(Module, sym, {
      configurable: true,
      get: function() {
        abort(unexportedMessage(sym, isFSSybol));
      }
    });
  }
}

function unexportedRuntimeFunction(sym, isFSSybol) {
  if (!Object.getOwnPropertyDescriptor(Module, sym)) {
    Module[sym] = () => abort(unexportedMessage(sym, isFSSybol));
  }
}

// end include: runtime_debug.js
var tempRet0 = 0;
var setTempRet0 = (value) => { tempRet0 = value; };
var getTempRet0 = () => tempRet0;

```

```

// === Preamble library stuff ===

// Documentation for the public APIs defined in this file must be updated in:
//   site/source/docs/api_reference/preamble.js.rst
// A prebuilt local version of the documentation is available at:
//   site/build/text/docs/api_reference/preamble.js.txt
// You can also build docs locally as HTML or other formats in site/
// An online HTML version (which may be of a different version of Emscripten)
//   is up at http://kripken.github.io/emscripten-
site/docs/api_reference/preamble.js.html

var wasmBinary;
if (Module['wasmBinary']) wasmBinary =
Module['wasmBinary'];legacyModuleProp('wasmBinary', 'wasmBinary');
var noExitRuntime = Module['noExitRuntime'] || true;legacyModuleProp('noExitRuntime',
'noExitRuntime');

if (typeof WebAssembly != 'object') {
  abort('no native wasm support detected');
}

// include: runtime_safe_heap.js

// In MINIMAL_RUNTIME, setValue() and getValue() are only available when building with
// safe heap enabled, for heap safety checking.
// In traditional runtime, setValue() and getValue() are always available (although
// their use is highly discouraged due to perf penalties)

/** @param {number} ptr
 * @param {number} value
 * @param {string} type
 * @param {number|boolean=} noSafe */
function setValue(ptr, value, type = 'i8', noSafe) {
  if (type.charAt(type.length-1) === '*') type = 'i32';
  switch (type) {
    case 'i1': HEAP8[((ptr)>>0)] = value; break;
    case 'i8': HEAP8[((ptr)>>0)] = value; break;
    case 'i16': HEAP16[((ptr)>>1)] = value; break;
    case 'i32': HEAP32[((ptr)>>2)] = value; break;
    case 'i64': (tempI64 = [value>>>0,(tempDouble=value, (+(Math.abs(tempDouble))) >=
1.0 ? (tempDouble > 0.0 ? ((Math.min((+(Math.floor((tempDouble)/4294967296.0))), 4294967295.0))|0)>>>0 : (~(~((+(Math.ceil((tempDouble +
+((~(tempDouble))>>>0))/4294967296.0))))>>>0) : 0)],HEAP32[((ptr)>>2)] =
tempI64[0],HEAP32[((ptr)+(4))>>2]) = tempI64[1]); break;
    case 'float': HEAPF32[((ptr)>>2)] = value; break;
    case 'double': HEAPF64[((ptr)>>3)] = value; break;
    default: abort('invalid type for setValue: ' + type);
  }
}

/** @param {number} ptr
 * @param {string} type

```

```

@param {number|boolean=} noSafe */
function getValue(ptr, type = 'i8', noSafe) {
  if (type.charAt(type.length-1) === '*') type = 'i32';
  switch (type) {
    case 'i1': return HEAP8[((ptr)>>0)];
    case 'i8': return HEAP8[((ptr)>>0)];
    case 'i16': return HEAP16[((ptr)>>1)];
    case 'i32': return HEAP32[((ptr)>>2)];
    case 'i64': return HEAP32[((ptr)>>2)];
    case 'float': return HEAPF32[((ptr)>>2)];
    case 'double': return Number(HEAPF64[((ptr)>>3)]);
    default: abort('invalid type for getValue: ' + type);
  }
  return null;
}

// end include: runtime_safe_heap.js
// Wasm globals

var wasmMemory;

//=====
// Runtime essentials
=====

// whether we are quitting the application. no code should run after this.
// set in exit() and abort()
var ABORT = false;

// set by exit() and abort(). Passed to 'onExit' handler.
// NOTE: This is also used as the process return code in shell environments
// but only when noExitRuntime is false.
var EXITSTATUS;

/** @type {function(*, string=)} */
function assert(condition, text) {
  if (!condition) {
    abort('Assertion failed' + (text ? ': ' + text : ''));
  }
}

// Returns the C function with a specified identifier (for C++, you need to do manual
// name mangling)
function getCFunc(ident) {
  var func = Module['_' + ident]; // closure exported function
  assert(func, 'Cannot call unknown function ' + ident + ', make sure it is exported');
  return func;
}

// C calling interface.
/** @param {string|null=} returnType
  @param {Array=} argTypes
  @param {Arguments|Array=} args
  @param {Object=} opts */
function ccall(ident, returnType, argTypes, args, opts) {

```

```

// For fast lookup of conversion functions
var toC = {
  'string': function(str) {
    var ret = 0;
    if (str === null && str !== undefined && str !== 0) { // null string
      // at most 4 bytes per UTF-8 code point, +1 for the trailing '\0'
      var len = (str.length << 2) + 1;
      ret = stackAlloc(len);
      stringToUTF8(str, ret, len);
    }
    return ret;
  },
  'array': function(arr) {
    var ret = stackAlloc(arr.length);
    writeArrayToMemory(arr, ret);
    return ret;
  }
};

function convertReturnValue(ret) {
  if (returnType === 'string') return UTF8ToString(ret);
  if (returnType === 'boolean') return Boolean(ret);
  return ret;
}

var func = getCFunc(ident);
var cArgs = [];
var stack = 0;
assert(returnType !== 'array', 'Return type should not be "array".');
if (args) {
  for (var i = 0; i < args.length; i++) {
    var converter = toC[argTypes[i]];
    if (converter) {
      if (stack === 0) stack = stackSave();
      cArgs[i] = converter(args[i]);
    } else {
      cArgs[i] = args[i];
    }
  }
}
var ret = func.apply(null, cArgs);
function onDone(ret) {
  if (stack !== 0) stackRestore(stack);
  return convertReturnValue(ret);
}

ret = onDone(ret);
return ret;
}

/** @param {string=} returnType
 * @param {Array=} argTypes
 * @param {Object=} opts */
function cwrap(ident, returnType, argTypes, opts) {
  return function() {

```

```

        return ccall(ident, returnType, argTypes, arguments, opts);
    }
}

// We used to include malloc/free by default in the past. Show a helpful error in
// builds with assertions.

// include: runtime_legacy.js

var ALLOC_NORMAL = 0; // Tries to use _malloc()
var ALLOC_STACK = 1; // Lives for the duration of the current function call

/***
 * allocate(): This function is no longer used by emscripten but is kept around to
 * avoid
 *           breaking external users.
 *           You should normally not use allocate(), and instead allocate
 *           memory using _malloc()/stackAlloc(), initialize it with
 *           setValue(), and so forth.
 * @param {(Uint8Array|Array<number>)} slab: An array of data.
 * @param {number=} allocator : How to allocate memory, see ALLOC_*
 */
function allocate(slab, allocator) {
    var ret;
    assert(typeof allocator == 'number', 'allocate no longer takes a type argument')
    assert(typeof slab != 'number', 'allocate no longer takes a number as arg0')

    if (allocator == ALLOC_STACK) {
        ret = stackAlloc(slab.length);
    } else {
        ret = _malloc(slab.length);
    }

    if (!slab.subarray && !slab.slice) {
        slab = new Uint8Array(slab);
    }
    HEAPU8.set(slab, ret);
    return ret;
}

// end include: runtime_legacy.js
// include: runtime_strings.js

// runtime_strings.js: Strings related runtime functions that are part of both
MINIMAL_RUNTIME and regular runtime.

var UTF8Decoder = typeof TextDecoder != 'undefined' ? new TextDecoder('utf8') :
undefined;

// Given a pointer 'ptr' to a null-terminated UTF8-encoded string in the given array
that contains uint8 values, returns
// a copy of that string as a Javascript String object.
/***

```

```

* heapOrArray is either a regular array, or a JavaScript typed array view.
* @param {number} idx
* @param {number=} maxBytesToRead
* @return {string}
*/
function UTF8ArrayToString(heapOrArray, idx, maxBytesToRead) {
    var endIdx = idx + maxBytesToRead;
    var endPtr = idx;
    // TextDecoder needs to know the byte length in advance, it doesn't stop on null
    // terminator by itself.
    // Also, use the length info to avoid running tiny strings through TextDecoder, since
    .subarray() allocates garbage.
    // (As a tiny code save trick, compare endPtr against endIdx using a negation, so
    that undefined means Infinity)
    while (heapOrArray[endPtr] && !(endPtr >= endIdx)) ++endPtr;

    if (endPtr - idx > 16 && heapOrArray.buffer && UTF8Decoder) {
        return UTF8Decoder.decode(heapOrArray.subarray(idx, endPtr));
    } else {
        var str = '';
        // If building with TextDecoder, we have already computed the string length above,
        so test loop end condition against that
        while (idx < endPtr) {
            // For UTF8 byte structure, see:
            // http://en.wikipedia.org/wiki/UTF-8#Description
            // https://www.ietf.org/rfc/rfc2279.txt
            // https://tools.ietf.org/html/rfc3629
            var u0 = heapOrArray[idx++];
            if (!(u0 & 0x80)) { str += String.fromCharCode(u0); continue; }
            var u1 = heapOrArray[idx++] & 63;
            if ((u0 & 0xE0) == 0xC0) { str += String.fromCharCode(((u0 & 31) << 6) | u1);
        continue; }
            var u2 = heapOrArray[idx++] & 63;
            if ((u0 & 0xF0) == 0x80) {
                u0 = ((u0 & 15) << 12) | (u1 << 6) | u2;
            } else {
                if ((u0 & 0xF8) != 0xF0) warnOnce('Invalid UTF-8 leading byte 0x' +
u0.toString(16) + ' encountered when deserializing a UTF-8 string in wasm memory to a
JS string!');
                u0 = ((u0 & 7) << 18) | (u1 << 12) | (u2 << 6) | (heapOrArray[idx++] & 63);
            }
            if (u0 < 0x10000) {
                str += String.fromCharCode(u0);
            } else {
                var ch = u0 - 0x10000;
                str += String.fromCharCode(0xD800 | (ch >> 10), 0xDC00 | (ch & 0x3FF));
            }
        }
        return str;
    }
}

// Given a pointer 'ptr' to a null-terminated UTF8-encoded string in the emscripten
// HEAP, returns a

```

```

// copy of that string as a Javascript String object.
// maxBytesToRead: an optional length that specifies the maximum number of bytes to
read. You can omit
//           this parameter to scan the string until the first \0 byte. If
maxBytesToRead is
//           passed, and the string at [ptr, ptr+maxBytesToRead[ contains a null
byte in the
//           middle, then the string will cut short at that byte index (i.e.
maxBytesToRead will
//           not produce a string of exact length [ptr, ptr+maxBytesToRead[])
//           N.B. mixing frequent uses of UTF8ToString() with and without
maxBytesToRead may
//           throw JS JIT optimizations off, so it is worth to consider
consistently using one
//           style or the other.
/***
 * @param {number} ptr
 * @param {number=} maxBytesToRead
 * @return {string}
 */
function UTF8ToString(ptr, maxBytesToRead) {
;
  return ptr ? UTF8ArrayToString(HEAPU8, ptr, maxBytesToRead) : '';
}

// Copies the given Javascript String object 'str' to the given byte array at address
// 'outIdx',
// encoded in UTF8 form and null-terminated. The copy will require at most
// str.length*4+1 bytes of space in the HEAP.
// Use the function lengthBytesUTF8 to compute the exact number of bytes (excluding
// null terminator) that this function will write.
// Parameters:
//   str: the Javascript string to copy.
//   heap: the array to copy to. Each index in this array is assumed to be one 8-byte
element.
//   outIdx: The starting offset in the array to begin the copying.
//   maxBytesToWrite: The maximum number of bytes this function can write to the array.
//                   This count should include the null terminator,
//                   i.e. if maxBytesToWrite=1, only the null terminator will be
written and nothing else.
//                   maxBytesToWrite=0 does not write any bytes to the output, not
even the null terminator.
// Returns the number of bytes written, EXCLUDING the null terminator.

function stringToUTF8Array(str, heap, outIdx, maxBytesToWrite) {
  if (!(maxBytesToWrite > 0)) // Parameter maxBytesToWrite is not optional. Negative
values, 0, null, undefined and false each don't write out any bytes.
  return 0;

  var startIdx = outIdx;
  var endIdx = outIdx + maxBytesToWrite - 1; // -1 for string null terminator.
  for (var i = 0; i < str.length; ++i) {
    // Gotcha: charCodeAt returns a 16-bit word that is a UTF-16 encoded code unit, not
a Unicode code point of the character! So decode UTF16->UTF32->UTF8.
    // See http://unicode.org/faq/utf_bom.html#utf16-3

```

```

// For UTF8 byte structure, see http://en.wikipedia.org/wiki/UTF-8#Description and
// https://www.ietf.org/rfc/rfc2279.txt and https://tools.ietf.org/html/rfc3629
var u = str.charCodeAt(i); // possibly a lead surrogate
if (u >= 0xD800 && u <= 0xDFFF) {
    var u1 = str.charCodeAt(++i);
    u = 0x10000 + ((u & 0x3FF) << 10) | (u1 & 0x3FF);
}
if (u <= 0x7F) {
    if (outIdx >= endIdx) break;
    heap[outIdx++] = u;
} else if (u <= 0x7FF) {
    if (outIdx + 1 >= endIdx) break;
    heap[outIdx++] = 0xC0 | (u >> 6);
    heap[outIdx++] = 0x80 | (u & 63);
} else if (u <= 0xFFFF) {
    if (outIdx + 2 >= endIdx) break;
    heap[outIdx++] = 0xE0 | (u >> 12);
    heap[outIdx++] = 0x80 | ((u >> 6) & 63);
    heap[outIdx++] = 0x80 | (u & 63);
} else {
    if (outIdx + 3 >= endIdx) break;
    if (u > 0x10FFFF) warnOnce('Invalid Unicode code point 0x' + u.toString(16) +
encountered when serializing a JS string to a UTF-8 string in wasm memory! (Valid
unicode code points should be in range 0-0x10FFFF).');
    heap[outIdx++] = 0xF0 | (u >> 18);
    heap[outIdx++] = 0x80 | ((u >> 12) & 63);
    heap[outIdx++] = 0x80 | ((u >> 6) & 63);
    heap[outIdx++] = 0x80 | (u & 63);
}
// Null-terminate the pointer to the buffer.
heap[outIdx] = 0;
return outIdx - startIdx;
}

// Copies the given Javascript String object 'str' to the emscripten HEAP at address
// 'outPtr',
// null-terminated and encoded in UTF8 form. The copy will require at most
// str.length*4+1 bytes of space in the HEAP.
// Use the function lengthBytesUTF8 to compute the exact number of bytes (excluding
// null terminator) that this function will write.
// Returns the number of bytes written, EXCLUDING the null terminator.

function stringToUTF8(str, outPtr, maxBytesToWrite) {
    assert(typeof maxBytesToWrite == 'number', 'stringToUTF8(str, outPtr,
maxBytesToWrite) is missing the third parameter that specifies the length of the output
buffer!');
    return stringToUTF8Array(str, HEAPU8,outPtr, maxBytesToWrite);
}

// Returns the number of bytes the given Javascript string takes if encoded as a UTF8
// byte array, EXCLUDING the null terminator byte.
function lengthBytesUTF8(str) {
    var len = 0;
    for (var i = 0; i < str.length; ++i) {

```

```

// Gotcha: charCodeAt returns a 16-bit word that is a UTF-16 encoded code unit, not
// a Unicode code point of the character! So decode UTF16->UTF32->UTF8.
// See http://unicode.org/faq/utf_bom.html#utf16-3
var u = str.charCodeAt(i); // possibly a lead surrogate
if (u >= 0xD800 && u <= 0xDFFF) u = 0x10000 + ((u & 0x3FF) << 10) +
(str.charCodeAt(++i) & 0x3FF);
if (u <= 0x7F) ++len;
else if (u <= 0x7FF) len += 2;
else if (u <= 0xFFFF) len += 3;
else len += 4;
}
return len;
}

// end include: runtime_strings.js
// include: runtime_strings_extra.js

// runtime_strings_extra.js: Strings related runtime functions that are available only
in regular runtime.

// Given a pointer 'ptr' to a null-terminated ASCII-encoded string in the emscripten
HEAP, returns
// a copy of that string as a Javascript String object.

function AsciiToString(ptr) {
    var str = '';
    while (1) {
        var ch = HEAPU8[((ptr++)>>0)];
        if (!ch) return str;
        str += String.fromCharCode(ch);
    }
}

// Copies the given Javascript String object 'str' to the emscripten HEAP at address
'outPtr',
// null-terminated and encoded in ASCII form. The copy will require at most
str.length+1 bytes of space in the HEAP.

function stringToAscii(str, outPtr) {
    return writeAsciiToMemory(str, outPtr, false);
}

// Given a pointer 'ptr' to a null-terminated UTF16LE-encoded string in the emscripten
HEAP, returns
// a copy of that string as a Javascript String object.

var UTF16Decoder = typeof TextDecoder != 'undefined' ? new TextDecoder('utf-16le') :
undefined;

function UTF16ToString(ptr, maxBytesToRead) {
    assert(ptr % 2 == 0, 'Pointer passed to UTF16ToString must be aligned to two
bytes!');
    var endPtr = ptr;
    // TextDecoder needs to know the byte length in advance, it doesn't stop on null
terminator by itself.
}

```

```

// Also, use the length info to avoid running tiny strings through TextDecoder, since
.subarray() allocates garbage.
var idx = endPtr >> 1;
var maxIdx = idx + maxBytesToRead / 2;
// If maxBytesToRead is not passed explicitly, it will be undefined, and this
// will always evaluate to true. This saves on code size.
while (!(idx >= maxIdx) && HEAPU16[idx]) ++idx;
endPtr = idx << 1;

if (endPtr - ptr > 32 && UTF16Decoder) {
    return UTF16Decoder.decode(HEAPU8.subarray(ptr, endPtr));
} else {
    var str = '';

    // If maxBytesToRead is not passed explicitly, it will be undefined, and the for-
    loop's condition
    // will always evaluate to true. The loop is then terminated on the first null
    char.
    for (var i = 0; !(i >= maxBytesToRead / 2); ++i) {
        var codeUnit = HEAP16[((ptr)+(i*2))>>1];
        if (codeUnit == 0) break;
        // fromCharCode constructs a character from a UTF-16 code unit, so we can pass
        the UTF16 string right through.
        str += String.fromCharCode(codeUnit);
    }

    return str;
}
}

// Copies the given Javascript String object 'str' to the emscripten HEAP at address
'outPtr',
// null-terminated and encoded in UTF16 form. The copy will require at most
str.length*4+2 bytes of space in the HEAP.
// Use the function lengthBytesUTF16() to compute the exact number of bytes (excluding
null terminator) that this function will write.
// Parameters:
//   str: the Javascript string to copy.
//   outPtr: Byte address in Emscripten HEAP where to write the string to.
//   maxBytesToWrite: The maximum number of bytes this function can write to the array.
This count should include the null
//           terminator, i.e. if maxBytesToWrite=2, only the null terminator
will be written and nothing else.
//           maxBytesToWrite<2 does not write any bytes to the output, not
even the null terminator.
// Returns the number of bytes written, EXCLUDING the null terminator.

function stringToUTF16(str, outPtr, maxBytesToWrite) {
    assert(outPtr % 2 == 0, 'Pointer passed to stringToUTF16 must be aligned to two
bytes!');
    assert(typeof maxBytesToWrite == 'number', 'stringToUTF16(str, outPtr,
maxBytesToWrite) is missing the third parameter that specifies the length of the output
buffer!');
    // Backwards compatibility: if max bytes is not specified, assume unsafe unbounded
write is allowed.
    if (maxBytesToWrite === undefined) {

```

```

        maxBytesToWrite = 0x7FFFFFFF;
    }
    if (maxBytesToWrite < 2) return 0;
    maxBytesToWrite -= 2; // Null terminator.
    var startPtr = outPtr;
    var numCharsToWrite = (maxBytesToWrite < str.length*2) ? (maxBytesToWrite / 2) :
str.length;
    for (var i = 0; i < numCharsToWrite; ++i) {
        // charCodeAt returns a UTF-16 encoded code unit, so it can be directly written to
        // the HEAP.
        var codeUnit = str.charCodeAt(i); // possibly a lead surrogate
        HEAP16[((outPtr)>>1)] = codeUnit;
        outPtr += 2;
    }
    // Null-terminate the pointer to the HEAP.
    HEAP16[((outPtr)>>1)] = 0;
    return outPtr - startPtr;
}

// Returns the number of bytes the given Javascript string takes if encoded as a UTF16
// byte array, EXCLUDING the null terminator byte.

function lengthBytesUTF16(str) {
    return str.length*2;
}

function UTF32ToString(ptr, maxBytesToRead) {
    assert(ptr % 4 == 0, 'Pointer passed to UTF32ToString must be aligned to four
bytes!');
    var i = 0;

    var str = '';
    // If maxBytesToRead is not passed explicitly, it will be undefined, and this
    // will always evaluate to true. This saves on code size.
    while (!(i >= maxBytesToRead / 4)) {
        var utf32 = HEAP32[((ptr)+(i*4))>>2];
        if (utf32 == 0) break;
        ++i;
        // Gotcha: fromCharCode constructs a character from a UTF-16 encoded code (pair),
        // not from a Unicode code point! So encode the code point to UTF-16 for constructing.
        // See http://unicode.org/faq/utf_bom.html#utf16-3
        if (utf32 >= 0x10000) {
            var ch = utf32 - 0x10000;
            str += String.fromCharCode(0xD800 | (ch >> 10), 0xDC00 | (ch & 0x3FF));
        } else {
            str += String.fromCharCode(utf32);
        }
    }
    return str;
}

// Copies the given Javascript String object 'str' to the emscripten HEAP at address
// 'outPtr',
// null-terminated and encoded in UTF32 form. The copy will require at most
// str.length*4+4 bytes of space in the HEAP.

```

```

// Use the function lengthBytesUTF32() to compute the exact number of bytes (excluding
// null terminator) that this function will write.
// Parameters:
//   str: the Javascript string to copy.
//   outPtr: Byte address in Emscripten HEAP where to write the string to.
//   maxBytesToWrite: The maximum number of bytes this function can write to the array.
This count should include the null
//           terminator, i.e. if maxBytesToWrite=4, only the null terminator
will be written and nothing else.
//           maxBytesToWrite<4 does not write any bytes to the output, not
even the null terminator.
// Returns the number of bytes written, EXCLUDING the null terminator.

function stringToUTF32(str, outPtr, maxBytesToWrite) {
    assert(outPtr % 4 == 0, 'Pointer passed to stringToUTF32 must be aligned to four
bytes!');
    assert(typeof maxBytesToWrite == 'number', 'stringToUTF32(str, outPtr,
maxBytesToWrite) is missing the third parameter that specifies the length of the output
buffer!');
    // Backwards compatibility: if max bytes is not specified, assume unsafe unbounded
write is allowed.
    if (maxBytesToWrite === undefined) {
        maxBytesToWrite = 0x7FFFFFFF;
    }
    if (maxBytesToWrite < 4) return 0;
    var startPtr = outPtr;
    var endPtr = startPtr + maxBytesToWrite - 4;
    for (var i = 0; i < str.length; ++i) {
        // Gotcha: charCodeAt returns a 16-bit word that is a UTF-16 encoded code unit, not
a Unicode code point of the character! We must decode the string to UTF-32 to the heap.
        // See http://unicode.org/faq/utf_bom.html#utf16-3
        var codeUnit = str.charCodeAt(i); // possibly a lead surrogate
        if (codeUnit >= 0xD800 && codeUnit <= 0xDFFF) {
            var trailSurrogate = str.charCodeAt(++i);
            codeUnit = 0x10000 + ((codeUnit & 0x3FF) << 10) | (trailSurrogate & 0x3FF);
        }
        HEAP32[((outPtr)>>2)] = codeUnit;
        outPtr += 4;
        if (outPtr + 4 > endPtr) break;
    }
    // Null-terminate the pointer to the HEAP.
    HEAP32[((outPtr)>>2)] = 0;
    return outPtr - startPtr;
}

// Returns the number of bytes the given Javascript string takes if encoded as a UTF16
byte array, EXCLUDING the null terminator byte.

function lengthBytesUTF32(str) {
    var len = 0;
    for (var i = 0; i < str.length; ++i) {
        // Gotcha: charCodeAt returns a 16-bit word that is a UTF-16 encoded code unit, not
a Unicode code point of the character! We must decode the string to UTF-32 to the heap.
        // See http://unicode.org/faq/utf_bom.html#utf16-3

```

```

    var codeUnit = str.charCodeAt(i);
    if (codeUnit >= 0xD800 && codeUnit <= 0xDFFF) ++i; // possibly a lead surrogate, so
skip over the tail surrogate.
    len += 4;
}

return len;
}

// Allocate heap space for a JS string, and write it there.
// It is the responsibility of the caller to free() that memory.
function allocateUTF8(str) {
    var size = lengthBytesUTF8(str) + 1;
    var ret = _malloc(size);
    if (ret) stringToUTF8Array(str, HEAP8, ret, size);
    return ret;
}

// Allocate stack space for a JS string, and write it there.
function allocateUTF8OnStack(str) {
    var size = lengthBytesUTF8(str) + 1;
    var ret = stackAlloc(size);
    stringToUTF8Array(str, HEAP8, ret, size);
    return ret;
}

// Deprecated: This function should not be called because it is unsafe and does not
provide
// a maximum length limit of how many bytes it is allowed to write. Prefer calling the
// function stringToUTF8Array() instead, which takes in a maximum length that can be
used
// to be secure from out of bounds writes.
/** @deprecated
 * @param {boolean=} dontAddNull */
function writeStringToMemory(string, buffer, dontAddNull) {
    warnOnce('writeStringToMemory is deprecated and should not be called! Use
stringToUTF8() instead!');

    var /** @type {number} */ lastChar, /** @type {number} */ end;
    if (dontAddNull) {
        // stringToUTF8Array always appends null. If we don't want to do that, remember the
        // character that existed at the location where the null will be placed, and
restore
        // that after the write (below).
        end = buffer + lengthBytesUTF8(string);
        lastChar = HEAP8[end];
    }
    stringToUTF8(string, buffer, Infinity);
    if (dontAddNull) HEAP8[end] = lastChar; // Restore the value under the null
character.
}

function writeArrayToMemory(array, buffer) {
    assert(array.length >= 0, 'writeArrayToMemory array must have a length (should be an
array or typed array)')
}

```

```

    HEAP8.set(array, buffer);
}

/** @param {boolean=} dontAddNull */
function writeAsciiToMemory(str, buffer, dontAddNull) {
    for (var i = 0; i < str.length; ++i) {
        assert(str.charCodeAt(i) === (str.charCodeAt(i) & 0xff));
        HEAP8[((buffer++)>>0)] = str.charCodeAt(i);
    }
    // Null-terminate the pointer to the HEAP.
    if (!dontAddNull) HEAP8[((buffer)>>0)] = 0;
}

// end include: runtime_strings_extra.js
// Memory management

var HEAP,
/** @type {!ArrayBuffer} */
    buffer,
/** @type {!Int8Array} */
    HEAP8,
/** @type {!Uint8Array} */
    HEAPU8,
/** @type {!Int16Array} */
    HEAP16,
/** @type {!Uint16Array} */
    HEAPU16,
/** @type {!Int32Array} */
    HEAP32,
/** @type {!Uint32Array} */
    HEAPU32,
/** @type {!Float32Array} */
    HEAPF32,
/** @type {!Float64Array} */
    HEAPF64;

function updateGlobalBufferAndViews(buf) {
    buffer = buf;
    Module['HEAP8'] = HEAP8 = new Int8Array(buf);
    Module['HEAP16'] = HEAP16 = new Int16Array(buf);
    Module['HEAP32'] = HEAP32 = new Int32Array(buf);
    Module['HEAPU8'] = HEAPU8 = new Uint8Array(buf);
    Module['HEAPU16'] = HEAPU16 = new Uint16Array(buf);
    Module['HEAPU32'] = HEAPU32 = new Uint32Array(buf);
    Module['HEAPF32'] = HEAPF32 = new Float32Array(buf);
    Module['HEAPF64'] = HEAPF64 = new Float64Array(buf);
}

var TOTAL_STACK = 5242880;
if (Module['TOTAL_STACK']) assert(TOTAL_STACK === Module['TOTAL_STACK'], 'the stack size can no longer be determined at runtime')

var INITIAL_MEMORY = Module['INITIAL_MEMORY'] ||
33554432; legacyModuleProp('INITIAL_MEMORY', 'INITIAL_MEMORY');

```

```

assert(INITIAL_MEMORY >= TOTAL_STACK, 'INITIAL_MEMORY should be larger than
TOTAL_STACK, was ' + INITIAL_MEMORY + '! (TOTAL_STACK=' + TOTAL_STACK + ')');

// check for full engine support (use string 'subarray' to avoid closure compiler
confusion)
assert(typeof Int32Array != 'undefined' && typeof Float64Array !== 'undefined' &&
Int32Array.prototype.subarray != undefined && Int32Array.prototype.set != undefined,
'JS engine does not provide full typed array support');

// If memory is defined in wasm, the user can't provide it.
assert(!Module['wasmMemory'], 'Use of `wasmMemory` detected. Use -s IMPORTED_MEMORY to
define wasmMemory externally');
assert(INITIAL_MEMORY == 33554432, 'Detected runtime INITIAL_MEMORY setting. Use -s
IMPORTED_MEMORY to define wasmMemory dynamically');

// include: runtime_init_table.js
// In regular non-RELOCATABLE mode the table is exported
// from the wasm module and this will be assigned once
// the exports are available.
var wasmTable;

// end include: runtime_init_table.js
// include: runtime_stack_check.js

// Initializes the stack cookie. Called at the startup of main and at the startup of
each thread in pthreads mode.
function writeStackCookie() {
    var max = _emscripten_stack_get_end();
    assert((max & 3) == 0);
    // The stack grow downwards towards _emscripten_stack_get_end.
    // We write cookies to the final two words in the stack and detect if they are
    // ever overwritten.
    HEAP32[((max)>>2)] = 0x2135467;
    HEAP32[((((max)+(4))>>2))] = 0x89BACDFE;
    // Also test the global address 0 for integrity.
    HEAP32[0] = 0x63736d65; /* 'emsc' */
}

function checkStackCookie() {
    if (ABORT) return;
    var max = _emscripten_stack_get_end();
    var cookie1 = HEAPU32[((max)>>2)];
    var cookie2 = HEAPU32[((((max)+(4))>>2))];
    if (cookie1 != 0x2135467 || cookie2 != 0x89BACDFE) {
        abort('Stack overflow! Stack cookie has been overwritten, expected hex dwords
0x89BACDFE and 0x2135467, but received 0x' + cookie2.toString(16) + ' 0x' +
cookie1.toString(16));
    }
    // Also test the global address 0 for integrity.
    if (HEAP32[0] !== 0x63736d65 /* 'emsc' */) abort('Runtime error: The application has
corrupted its heap memory area (address zero)!');
}

// end include: runtime_stack_check.js

```

```

// include: runtime_assertions.js

// Endianness check
(function() {
  var h16 = new Int16Array(1);
  var h8 = new Int8Array(h16.buffer);
  h16[0] = 0x6373;
  if (h8[0] !== 0x73 || h8[1] !== 0x63) throw 'Runtime error: expected the system to be
little-endian! (Run with -s SUPPORT_BIG_ENDIAN=1 to bypass)';
})();

// end include: runtime_assertions.js
var __ATPRERUN__ = []; // functions called before the runtime is initialized
var __ATINIT__ = []; // functions called during startup
var __ATMAIN__ = []; // functions called when main() is to be run
var __ATEXIT__ = []; // functions called during shutdown
var __ATPOSTRUN__ = []; // functions called after the main() is called

var runtimeInitialized = false;

function keepRuntimeAlive() {
  return noExitRuntime;
}

function preRun() {

  if (Module['preRun']) {
    if (typeof Module['preRun'] == 'function') Module['preRun'] = [Module['preRun']];
    while (Module['preRun'].length) {
      addOnPreRun(Module['preRun'].shift());
    }
  }

  callRuntimeCallbacks(__ATPRERUN__);
}

function initRuntime() {
  checkStackCookie();
  assert(!runtimeInitialized);
  runtimeInitialized = true;

  if (!Module["noFSInit"] && !FS.init.initialized)
    FS.init();
  FS.ignorePermissions = false;

  TTY.init();
  SOCKFS.root = FS.mount(SOCKFS, {}, null);
  callRuntimeCallbacks(__ATINIT__);
}

function preMain() {
  checkStackCookie();
}

```

```

    callRuntimeCallbacks(__ATMAIN__);
}

function postRun() {
    checkStackCookie();

    if (Module['postRun']) {
        if (typeof Module['postRun'] == 'function') Module['postRun'] =
[Module['postRun']];
        while (Module['postRun'].length) {
            addOnPostRun(Module['postRun'].shift());
        }
    }
    callRuntimeCallbacks(__ATPOSTRUN__);
}

function addOnPreRun(cb) {
    __ATPRERUN__.unshift(cb);
}

function addOnInit(cb) {
    __ATINIT__.unshift(cb);
}

function addOnPreMain(cb) {
    __ATMAIN__.unshift(cb);
}

function addOnExit(cb) {
}

function addOnPostRun(cb) {
    __ATPOSTRUN__.unshift(cb);
}

// include: runtime_math.js

// https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Math/imul
// https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Math/fround
// https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Math/clz32
// https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Math/trunc

assert(Math.imul, 'This browser does not support Math.imul(), build with LEGACY_VM_SUPPORT or POLYFILL_OLD_MATH_FUNCTIONS to add in a polyfill');
assert(Math.fround, 'This browser does not support Math.fround(), build with LEGACY_VM_SUPPORT or POLYFILL_OLD_MATH_FUNCTIONS to add in a polyfill');

```

```

assert(Math.clz32, 'This browser does not support Math.clz32(), build with
LEGACY_VM_SUPPORT or POLYFILL_OLD_MATH_FUNCTIONS to add in a polyfill');
assert(Math.trunc, 'This browser does not support Math.trunc(), build with
LEGACY_VM_SUPPORT or POLYFILL_OLD_MATH_FUNCTIONS to add in a polyfill');

// end include: runtime_math.js
// A counter of dependencies for calling run(). If we need to
// do asynchronous work before running, increment this and
// decrement it. Incrementing must happen in a place like
// Module.preRun (used by emcc to add file preloading).
// Note that you can add dependencies in preRun, even though
// it happens right before run - run will be postponed until
// the dependencies are met.
var runDependencies = 0;
var runDependencyWatcher = null;
var dependenciesFulfilled = null; // overridden to take different actions when all run
dependencies are fulfilled
var runDependencyTracking = {};

function getUniqueRunDependency(id) {
    var orig = id;
    while (1) {
        if (!runDependencyTracking[id]) return id;
        id = orig + Math.random();
    }
}

function addRunDependency(id) {
    runDependencies++;

    if (Module['monitorRunDependencies']) {
        Module['monitorRunDependencies'](runDependencies);
    }

    if (id) {
        assert(!runDependencyTracking[id]);
        runDependencyTracking[id] = 1;
        if (runDependencyWatcher === null && typeof setInterval !== 'undefined') {
            // Check for missing dependencies every few seconds
            runDependencyWatcher = setInterval(function() {
                if (ABORT) {
                    clearInterval(runDependencyWatcher);
                    runDependencyWatcher = null;
                    return;
                }
                var shown = false;
                for (var dep in runDependencyTracking) {
                    if (!shown) {
                        shown = true;
                        err('still waiting on run dependencies:');
                    }
                    err('dependency: ' + dep);
                }
                if (shown) {
                    err('(end of list)');
                }
            });
        }
    }
}

```

```

        }
    }, 10000);
}
} else {
    err('warning: run dependency added without ID');
}
}

function removeRunDependency(id) {
    runDependencies--;
    if (Module['monitorRunDependencies']) {
        Module['monitorRunDependencies'](runDependencies);
    }
    if (id) {
        assert(runDependencyTracking[id]);
        delete runDependencyTracking[id];
    } else {
        err('warning: run dependency removed without ID');
    }
    if (runDependencies == 0) {
        if (runDependencyWatcher !== null) {
            clearInterval(runDependencyWatcher);
            runDependencyWatcher = null;
        }
        if (dependenciesFulfilled) {
            var callback = dependenciesFulfilled;
            dependenciesFulfilled = null;
            callback(); // can add another dependenciesFulfilled
        }
    }
}

Module["preloadedImages"] = {};// maps url to image data
Module["preloadedAudios"] = {};// maps url to audio data

/** @param {string|number=} what */
function abort(what) {
    if (Module['onAbort']) {
        Module['onAbort'](what);
    }
    what = 'Aborted(' + what + ')';
    // TODO(sbc): Should we remove printing and leave it up to whoever
    // catches the exception?
    err(what);
    ABORT = true;
    EXITSTATUS = 1;
    // Use a wasm runtime error, because a JS error might be seen as a foreign
    // exception, which means we'd run destructors on it. We need the error to

```

```

// simply make the program stop.

// Suppress closure compiler warning here. Closure compiler's builtin extern
// defintion for WebAssembly.RuntimeError claims it takes no arguments even
// though it can.
// TODO(https://github.com/google/closure-compiler/pull/3913): Remove if/when
upstream closure gets fixed.

/** @suppress {checkTypes} */
var e = new WebAssembly.RuntimeError(what);

readyPromiseReject(e);
// Throw the error whether or not MODULARIZE is set because abort is used
// in code paths apart from instantiation where an exception is expected
// to be thrown when abort is called.
throw e;
}

// {{MEM_INITIALIZER}}

// include: memoryprofiler.js

// end include: memoryprofiler.js
// include: URIUtils.js

// Prefix of data URIs emitted by SINGLE_FILE and related options.
var dataURIPrefix = 'data:application/octet-stream;base64,';

// Indicates whether filename is a base64 data URI.
function isDataURI(filename) {
    // Prefix of data URIs emitted by SINGLE_FILE and related options.
    return filename.startsWith(dataURIPrefix);
}

// Indicates whether filename is delivered via file protocol (as opposed to http/https)
function isFileURI(filename) {
    return filename.startsWith('file://');
}

// end include: URIUtils.js
/** @param {boolean=} fixedasm */
function createExportWrapper(name, fixedasm) {
    return function() {
        var displayName = name;
        var asm = fixedasm;
        if (!fixedasm) {
            asm = Module['asm'];
        }
        assert(runtimeInitialized, 'native function `' + displayName + '` called before
runtime initialization');
        if (!asm[name]) {
            assert(asm[name], 'exported native function `' + displayName + '` not found');
        }
    }
}

```

```

        return asm[name].apply(null, arguments);
    };
}

var wasmBinaryFile;
wasmBinaryFile = 'build.wasm';
if (!isDataURI(wasmBinaryFile)) {
    wasmBinaryFile = locateFile(wasmBinaryFile);
}

function getBinary(file) {
    try {
        if (file == wasmBinaryFile && wasmBinary) {
            return new Uint8Array(wasmBinary);
        }
        if (readBinary) {
            return readBinary(file);
        } else {
            throw "both async and sync fetching of the wasm failed";
        }
    }
    catch (err) {
        abort(err);
    }
}

function getBinaryPromise() {
    // If we don't have the binary yet, try to load it asynchronously.
    // Fetch has some additional restrictions over XHR, like it can't be used on a
    file:// url.
    // See https://github.com/github/fetch/pull/92#issuecomment-140665932
    // Cordova or Electron apps are typically loaded from a file:// url.
    // So use fetch if it is available and the url is not a file, otherwise fall back to
    XHR.
    if (!wasmBinary && (ENVIRONMENT_IS_WEB || ENVIRONMENT_IS_WORKER)) {
        if (typeof fetch == 'function'
            && !isFileURI(wasmBinaryFile)
        ) {
            return fetch(wasmBinaryFile, { credentials: 'same-origin'
        }).then(function(response) {
            if (!response['ok']) {
                throw "failed to load wasm binary file at '" + wasmBinaryFile + "'";
            }
            return response['arrayBuffer']();
        }).catch(function () {
            return getBinary(wasmBinaryFile);
        });
    }
    else {
        if (readAsync) {
            // fetch is not available or url is file => try XHR (readAsync uses XHR
internally)
            return new Promise(function(resolve, reject) {
                readAsync(wasmBinaryFile, function(response) { resolve(new Uint8Array/** @type{!ArrayBuffer} */(response)) }, reject)
            })
        }
    }
}

```

```

        });
    }
}

// Otherwise, getBinary should be able to get it synchronously
return Promise.resolve().then(function() { return getBinary(wasmBinaryFile); });
}

// Create the wasm instance.
// Receives the wasm imports, returns the exports.
function createWasm() {
    // prepare imports
    var info = {
        'env': asmLibraryArg,
        'wasi_snapshot_preview1': asmLibraryArg,
    };
    // Load the wasm module and create an instance of using native support in the JS
    // engine.
    // handle a generated wasm instance, receiving its exports and
    // performing other necessary setup
    /** @param {WebAssembly.Module=} module */
    function receiveInstance(instance, module) {
        var exports = instance.exports;

        Module['asm'] = exports;

        wasmMemory = Module['asm']['memory'];
        assert(wasmMemory, "memory not found in wasm exports");
        // This assertion doesn't hold when emscripten is run in --post-link
        // mode.
        // TODO(sbc): Read INITIAL_MEMORY out of the wasm file in post-link mode.
        //assert(wasmMemory.buffer.byteLength === 33554432);
        updateGlobalBufferAndViews(wasmMemory.buffer);

        wasmTable = Module['asm'][ '__indirect_function_table'];
        assert(wasmTable, "table not found in wasm exports");

        addOnInit(Module['asm'][ '__wasm_call_ctors']);

        removeRunDependency('wasm-instantiate');
    }
    // we can't run yet (except in a pthread, where we have a custom sync instantiator)
    addRunDependency('wasm-instantiate');

    // Prefer streaming instantiation if available.
    // Async compilation can be confusing when an error on the page overwrites Module
    // (for example, if the order of elements is wrong, and the one defining Module is
    // later), so we save Module and check it later.
    var trueModule = Module;
    function receiveInstantiationResult(result) {
        // 'result' is a ResultObject object which has both the module and instance.
        // receiveInstance() will swap in the exports (to Module.asm) so they can be called

```

```

    assert(Module === trueModule, 'the Module object should not be replaced during
async compilation - perhaps the order of HTML elements is wrong?');
    trueModule = null;
    // TODO: Due to Closure regression https://github.com/google/closure-
compiler/issues/3193, the above line no longer optimizes out down to the following
line.
    // When the regression is fixed, can restore the above USE_THREADS-enabled path.
    receiveInstance(result['instance']);
}

function instantiateArrayBuffer(receiver) {
    return getBinaryPromise().then(function(binary) {
        return WebAssembly.instantiate(binary, info);
    }).then(function(instance) {
        return instance;
    }).then(receiver, function(reason) {
        err('failed to asynchronously prepare wasm: ' + reason);

        // Warn on some common problems.
        if (isFileURI(wasmBinaryFile)) {
            err('warning: Loading from a file URI (' + wasmBinaryFile + ') is not supported
in most browsers. See https://emscripten.org/docs/getting_started/FAQ.html#how-do-i-
run-a-local-webserver-for-testing-why-does-my-program-stall-in-downloading-or-
preparing');
        }
        abort(reason);
    });
}

function instantiateAsync() {
    if (!wasmBinary &&
        typeof WebAssembly.instantiateStreaming == 'function' &&
        !isDataURI(wasmBinaryFile) &&
        // Don't use streaming for file:// delivered objects in a webview, fetch them
        // synchronously.
        !isFileURI(wasmBinaryFile) &&
        typeof fetch == 'function') {
        return fetch(wasmBinaryFile, { credentials: 'same-origin'
}).then(function(response) {
        // Suppress closure warning here since the upstream definition for
        // instantiateStreaming only allows Promise<Response> rather than
        // an actual Response.
        // TODO(https://github.com/google/closure-compiler/pull/3913): Remove if/when
        // upstream closure is fixed.
        /** @suppress {checkTypes} */
        var result = WebAssembly.instantiateStreaming(response, info);

        return result.then(
            receiveInstantiationResult,
            function(reason) {
                // We expect the most common failure cause to be a bad MIME type for the
                // binary,
                // in which case falling back to ArrayBuffer instantiation should work.
                err('wasm streaming compile failed: ' + reason);
                err('falling back to ArrayBuffer instantiation');
                return instantiateArrayBuffer(receiveInstantiationResult);
            });
    });
}

```

```

    });
} else {
    return instantiateArrayBuffer(receiveInstantiationResult);
}
}

// User shell pages can write their own Module.instantiateWasm = function(imports,
successCallback) callback
// to manually instantiate the Wasm module themselves. This allows pages to run the
instantiation parallel
// to any other async startup actions they are performing.
// Also pthreads and wasm workers initialize the wasm instance through this path.
if (Module['instantiateWasm']) {
    try {
        var exports = Module['instantiateWasm'](info, receiveInstance);
        return exports;
    } catch(e) {
        err('Module.instantiateWasm callback failed with error: ' + e);
        return false;
    }
}

// If instantiation fails, reject the module ready promise.
instantiateAsync().catch(readyPromiseReject);
return {}; // no exports yet; we'll fill them in later
}

// Globals used by JS i64 conversions (see makeSetValue)
var tempDouble;
var tempI64;

// === Body ===

var ASM_CONSTS = {
    3850588: function() {Module['emscripten_get_now_backup'] = performance.now;},
    3850643: function($0) {performance.now = function() { return $0; };},
    3850691: function($0) {performance.now = function() { return $0; };},
    3850739: function() {performance.now = Module['emscripten_get_now_backup'];},
    3850794: function() {return Module.webglContextAttributes.premultipliedAlpha;},
    3850855: function() {return Module.webglContextAttributes.preserveDrawingBuffer;},
    3850919: function() {return Module.webglContextAttributes.powerPreference;}
};

function callRuntimeCallbacks(callbacks) {
    while (callbacks.length > 0) {
        var callback = callbacks.shift();
        if (typeof callback == 'function') {
            callback(Module); // Pass the module as the first argument.
            continue;
        }
    }
}

```

```

var func = callback.func;
if (typeof func == 'number') {
    if (callback.arg === undefined) {
        // Run the wasm function ptr with signature 'v'. If no function
        // with such signature was exported, this call does not need
        // to be emitted (and would confuse Closure)
        (function() { dynCall_v.call(null, func); })();
    } else {
        // If any function with signature 'vi' was exported, run
        // the callback with that signature.
        (function(a1) { dynCall_vi.apply(null, [func, a1]); })(callback.arg);
    }
} else {
    func(callback.arg === undefined ? null : callback.arg);
}
}

function withStackSave(f) {
    var stack = stackSave();
    var ret = f();
    stackRestore(stack);
    return ret;
}
function demangle(func) {
    // If demangle has failed before, stop demangling any further function names
    // This avoids an infinite recursion with malloc()->abort()->stackTrace()-
>demangle()->malloc()->...
    demangle.recursionGuard = (demangle.recursionGuard|0)+1;
    if (demangle.recursionGuard > 1) return func;
    var __cxa_demangle_func = Module['__cxa_demangle'] || Module['__cxa_demangle'];
    assert(__cxa_demangle_func);
    return withStackSave(function() {
        try {
            var s = func;
            if (s.startsWith('__Z'))
                s = s.substr(1);
            var len = lengthBytesUTF8(s)+1;
            var buf = stackAlloc(len);
            stringToUTF8(s, buf, len);
            var status = stackAlloc(4);
            var ret = __cxa_demangle_func(buf, 0, 0, status);
            if (HEAP32[((status)>>2)] === 0 && ret) {
                return UTF8ToString(ret);
            }
            // otherwise, libcxxabi failed
        } catch(e) {
        } finally {
            free(ret);
            if (demangle.recursionGuard < 2) --demangle.recursionGuard;
        }
        // failure when using libcxxabi, don't demangle
        return func;
    });
}

```

```

function demangleAll(text) {
    var regex =
        /\b_Z[\w\d_]+/g;
    return text.replace(regex,
        function(x) {
            var y = demangle(x);
            return x === y ? x : (y + ' [' + x + ']');
        });
}

function dynCallLegacy(sig, ptr, args) {
    assert('dynCall_' + sig) in Module, 'bad function pointer type - no table for
sig \'' + sig + '\'';
    if (args && args.length) {
        // j (64-bit integer) must be passed in as two numbers [low 32, high 32].
        assert(args.length === sig.substring(1).replace(/j/g, '--').length);
    } else {
        assert(sig.length == 1);
    }
    var f = Module["dynCall_" + sig];
    return args && args.length ? f.apply(null, [ptr].concat(args)) : f.call(null,
ptr);
}

var wasmTableMirror = [];
function getWasmTableEntry(funcPtr) {
    var func = wasmTableMirror[funcPtr];
    if (!func) {
        if (funcPtr >= wasmTableMirror.length) wasmTableMirror.length = funcPtr + 1;
        wasmTableMirror[funcPtr] = func = wasmTable.get(funcPtr);
    }
    assert(wasmTable.get(funcPtr) == func, "JavaScript-side Wasm function table
mirror is out of date!");
    return func;
}
/** @param {Object=} args */
function dynCall(sig, ptr, args) {
    return dynCallLegacy(sig, ptr, args);
}

function handleException(e) {
    // Certain exception types we do not treat as errors since they are used for
    // internal control flow.
    // 1. ExitStatus, which is thrown by exit()
    // 2. "unwind", which is thrown by emscripten_unwind_to_js_event_loop() and
others
    //      that wish to return to JS event loop.
    if (e instanceof ExitStatus || e == 'unwind') {
        return EXITSTATUS;
    }
    quit_(1, e);
}

```

```

function jsStackTrace() {
    var error = new Error();
    if (!error.stack) {
        // IE10+ special cases: It does have callstack info, but it is only populated
        // if an Error object is thrown,
        // so try that as a special-case.
        try {
            throw new Error();
        } catch(e) {
            error = e;
        }
        if (!error.stack) {
            return '(no stack trace available)';
        }
    }
    return error.stack.toString();
}

function setWasmTableEntry(idx, func) {
    wasmTable.set(idx, func);
    wasmTableMirror[idx] = func;
}

function stackTrace() {
    var js = jsStackTrace();
    if (Module['extraStackTrace']) js += '\n' + Module['extraStackTrace']();
    return demangleAll(js);
}

function _GetJSMemoryInfo(totalJSptr, usedJSptr) {
    if (performance.memory) {
        HEAPF64[totalJSptr >> 3] = performance.memory.totalJSHeapSize;
        HEAPF64[usedJSptr >> 3] = performance.memory.usedJSHeapSize;
    } else {
        HEAPF64[totalJSptr >> 3] = NaN;
        HEAPF64[usedJSptr >> 3] = NaN;
    }
}

var JS_Accelerometer = null;

var JS_Accelerometer_callback = 0;
function JS_Accelerometer_IsRunning() {
    // Sensor is running if there is an activated new JS_Accelerometer; or the
    // JS_Accelerometer_callback is hooked up
    return (JS_Accelerometer && JS_Accelerometer.activated) ||
    (JS_Accelerometer_callback != 0);
}

var JS_Accelerometer_multiplier = 1;

var JS_Accelerometer_lastValue = {x:0,y:0,z:0};
function JS_Accelerometer_eventHandler() {
    // Record the last value for gravity computation
    JS_Accelerometer_lastValue =

```

```

        x: JS_Accelerometer.x * JS_Accelerometer_multiplier,
        y: JS_Accelerometer.y * JS_Accelerometer_multiplier,
        z: JS_Accelerometer.z * JS_Accelerometer_multiplier
    };
    if (JS_Accelerometer_callback != 0)
        dynCall_vfff(JS_Accelerometer_callback, JS_Accelerometer_lastValue.x,
JS_Accelerometer_lastValue.y, JS_Accelerometer_lastValue.z);
}

var JS_Accelerometer_frequencyRequest = 0;

var JS_Accelerometer_frequency = 0;

var JS_LinearAccelerationSensor_callback = 0;

var JS_GravitySensor_callback = 0;

var JS_Gyroscope_callback = 0;

function JS_ComputeGravity(accelerometerValue, linearAccelerationValue) {
    // On some Android devices, the linear acceleration direction is reversed
compared to its accelerometer
    // So, compute both the difference and sum (difference of the negative) and
return the one that's the smallest in magnitude
    var difference = {
        x: accelerometerValue.x - linearAccelerationValue.x,
        y: accelerometerValue.y - linearAccelerationValue.y,
        z: accelerometerValue.z - linearAccelerationValue.z
    };
    var differenceMagnitudeSq = difference.x*difference.x +
difference.y*difference.y + difference.z*difference.z;

    var sum = {
        x: accelerometerValue.x + linearAccelerationValue.x,
        y: accelerometerValue.y + linearAccelerationValue.y,
        z: accelerometerValue.z + linearAccelerationValue.z
    };
    var sumMagnitudeSq = sum.x*sum.x + sum.y*sum.y + sum.z*sum.z;

    return (differenceMagnitudeSq <= sumMagnitudeSq) ? difference : sum;
}
function JS_DeviceMotion_eventHandler(event) {
    // The accelerationIncludingGravity property is the amount of acceleration
recorded by the device, in meters per second squared (m/s2).
    // Its value is the sum of the acceleration of the device as induced by the
user and the acceleration caused by gravity.
    // Apply the JS_Accelerometer_multiplier to convert to g
    var accelerometerValue = {
        x: event.accelerationIncludingGravity.x * JS_Accelerometer_multiplier,
        y: event.accelerationIncludingGravity.y * JS_Accelerometer_multiplier,
        z: event.accelerationIncludingGravity.z * JS_Accelerometer_multiplier
    };
    if (JS_Accelerometer_callback != 0)
        dynCall_vfff(JS_Accelerometer_callback, accelerometerValue.x,
accelerometerValue.y, accelerometerValue.z);
}

```

```

        // The acceleration property is the amount of acceleration recorded by the
device, in meters per second squared (m/s2), compensated for gravity.
        // Apply the JS_Accelerometer_multiplier to convert to g
        var linearAccelerationValue = {
            x: event.acceleration.x * JS_Accelerometer_multiplier,
            y: event.acceleration.y * JS_Accelerometer_multiplier,
            z: event.acceleration.z * JS_Accelerometer_multiplier
        };
        if (JS_LinearAccelerationSensor_callback != 0)
            dynCall_vfff(JS_LinearAccelerationSensor_callback,
linearAccelerationValue.x, linearAccelerationValue.y, linearAccelerationValue.z);

        // Compute and raise the gravity sensor vector
        if (JS_GravitySensor_callback != 0) {
            assert(typeof GravitySensor === 'undefined');
            var gravityValue = JS_ComputeGravity(accelerometerValue,
linearAccelerationValue);
            dynCall_vfff(JS_GravitySensor_callback, gravityValue.x, gravityValue.y,
gravityValue.z);
        }

        // The rotationRate property describes the rotation rates of the device
around each of its axes (deg/s), but we want in radians/s so must scale
        // Note that the spec here has been updated so x,y,z axes are
alpha,beta,gamma.
        // Therefore the order of axes at https://developer.mozilla.org/en-
US/docs/Web/API/DeviceMotionEvent/rotationRate is incorrect
        //
        // There is a bug in Chrome < M66 where rotationRate values are not in deg/s
https://bugs.chromium.org/p/chromium/issues/detail?id=541607
        // But that version is too old to include a check here
        if (JS_Gyroscope_callback != 0) {
            var degToRad = Math.PI / 180;
            dynCall_vfff(JS_Gyroscope_callback, event.rotationRate.alpha * degToRad,
event.rotationRate.beta * degToRad, event.rotationRate.gamma * degToRad);
        }
    }

var JS_DeviceSensorPermissions = 0;
function JS_RequestDeviceSensorPermissions(permissions) {
    // iOS requires that we request permissions before using device sensor events
    if (permissions & 1/*DeviceOrientationEvent permission*/) {
        if (typeof DeviceOrientationEvent.requestPermission === 'function') {
            DeviceOrientationEvent.requestPermission()
                .then(function(permissionState) {
                    if (permissionState === 'granted') {
                        JS_DeviceSensorPermissions &= ~1; // Remove
DeviceOrientationEvent permission bit
                    } else {
                        warnOnce("DeviceOrientationEvent permission not
granted");
                    }
                })
                .catch(function(err) {

```

```

                // Permissions cannot be requested unless on a user
interaction (a touch event)
                    // So in this case set JS_DeviceSensorPermissions and we will
try again on a touch event
                        warnOnce(err);
                        JS_DeviceSensorPermissions |= 1/*DeviceOrientationEvent
permission*/;
                    });
                }
            }
        if (permissions & 2/*DeviceMotionEvent permission*/) {
            if (typeof DeviceMotionEvent.requestPermission === 'function') {
                DeviceMotionEvent.requestPermission()
                    .then(function(permissionState) {
                        if (permissionState === 'granted') {
                            JS_DeviceSensorPermissions &= ~2; // Remove
DeviceMotionEvent permission bit
                        } else {
                            warnOnce("DeviceMotionEvent permission not granted");
                        }
                    })
                    .catch(function(err) {
                        // Permissions cannot be requested unless on a user
interaction (a touch event)
                            // So in this case set JS_DeviceSensorPermissions and we will
try again on a touch event
                            warnOnce(err);
                            JS_DeviceSensorPermissions |= 2/*DeviceMotionEvent
permission*/;
                    });
                }
            }
        }
    }

function JS_DeviceMotion_add() {
    // Only add the event listener if we don't yet have any of the motion
callbacks set
    if (JS_Accelerometer_callback == 0 &&
        JS_LinearAccelerationSensor_callback == 0 &&
        JS_GravitySensor_callback == 0 &&
        JS_Gyroscope_callback == 0) {
        JS_RequestDeviceSensorPermissions(2/*DeviceMotionEvent permission*/);
        window.addEventListener('devicemotion', JS_DeviceMotion_eventHandler);
    }
}

function JS_DefineAccelerometerMultiplier() {
    // Earth's gravity in m/s^2, same as ASENSOR_STANDARD_GRAVITY
    var g = 9.80665;

    // Multiplier is 1/g to normalize acceleration
    // iOS has its direction opposite to Android and Windows (tested Surface Pro
tablet)
    // We include Macintosh in the test to capture Safari on iOS viewing in
Desktop mode (the default now on iPads)
}

```

```

    JS_Accelerometer_multiplier =
((iPhone|iPad|Macintosh)/i.test(navigator.userAgent)) ? 1/g : -1/g;
}
function JS_Accelerometer_Start(callback, frequency) {
// callback can be zero here when called via JS_GravitySensor_Start

JS_DefineAccelerometerMultiplier();

// If we don't have new sensor API, fallback to old DeviceMotionEvent
if (typeof Accelerometer === 'undefined') {
    JS_DeviceMotion_add(); // Must call before we set the callback
    if (callback != 0) JS_Accelerometer_callback = callback;
    return;
}

if (callback != 0) JS_Accelerometer_callback = callback;

function InitializeAccelerometer(frequency) {
    // Use device referenceFrame, since New Input System package does its own
compensation
    JS_Accelerometer = new Accelerometer({ frequency: frequency,
referenceFrame: 'device' });
    JS_Accelerometer.addEventListener('reading',
JS_Accelerometer_eventHandler);
    JS_Accelerometer.addEventListener('error', function(e) {
        // e.error could be DOMException: Could not connect to a sensor
        warnOnce((e.error) ? e.error : e);
    });
    JS_Accelerometer.start();
    JS_Accelerometer_frequency = frequency;
}

// If the sensor is already created, stop and re-create it with new frequency
if (JS_Accelerometer) {
    if (JS_Accelerometer_frequency != frequency) {
        JS_Accelerometer.stop();
        JS_Accelerometer.removeEventListener('reading',
JS_Accelerometer_eventHandler);
        InitializeAccelerometer(frequency);
    }
}
else if (JS_Accelerometer_frequencyRequest != 0) {
    // If the permissions promise is currently in progress, then note new
frequency only
    JS_Accelerometer_frequencyRequest = frequency;
}
else {
    JS_Accelerometer_frequencyRequest = frequency;

    // Request required permission for the Accelerometer
    navigator.permissions.query({name: 'accelerometer'})
        .then(function(result) {
            if (result.state === "granted") {
                InitializeAccelerometer(JS_Accelerometer_frequencyRequest);
            } else {
                warnOnce("No permission to use Accelerometer.");
            }
        })
}
}

```

```

        }
        JS_Accelerometer_frequencyRequest = 0;
    });
}
}

function JS_DeviceMotion_remove() {
    // If we've removed the last callback, remove the devicemotion event listener
    if (JS_Accelerometer_callback == 0 &&
        JS_LinearAccelerationSensor_callback == 0 &&
        JS_GravitySensor_callback == 0 &&
        JS_Gyroscope_callback == 0 ) {
        window.removeEventListener('devicemotion',
JS_DeviceOrientation_eventHandler);
    }
}
function _JS_Accelerometer_Stop() {
    if (JS_Accelerometer) {
        // Only actually stop the accelerometer if we don't need it to compute
gravity values
        if (typeof GravitySensor !== 'undefined' || JS_GravitySensor_callback ==
0) {
            JS_Accelerometer.stop();
            JS_Accelerometer.removeEventListener('reading',
JS_Accelerometer_eventHandler);
            JS_Accelerometer = null;
        }
        JS_Accelerometer_callback = 0;
        JS_Accelerometer_frequency = 0;
    }
    else if (JS_Accelerometer_callback != 0) {
        JS_Accelerometer_callback = 0;
        JS_DeviceMotion_remove();
    }
}

function _JS_Cursor_SetImage(ptr, length) {
    var binary = "";
    for (var i = 0; i < length; i++)
        binary += String.fromCharCode(HEAPU8[ptr + i]);
    Module.canvas.style.cursor = "url(data:image/cursor;base64," + btoa(binary) +
"),default";
}

function _JS_Cursor_SetShow(show) {
    Module.canvas.style.cursor = show ? "default" : "none";
}

function jsDomCssEscapeId(id) {
    // Use CSS Object Model to escape ID if feature is present
    if (typeof window.CSS !== "undefined" && typeof window.CSS.escape !==
"undefined") {
        return window.CSS.escape(id);
    }
}

```

```

        // Fallback: Escape special characters with RegExp. This handles most cases
        // but not all!
        return id.replace(/(\#|\.|\+|\[\[]|\|(\\)|\{|\})/g, "\\$1");
    }
}

function jsCanvasSelector() {
    // This lookup specifies the target canvas that different DOM
    // events are registered to, like keyboard and mouse events.
    // This requires that Module['canvas'] must have a CSS ID associated
    // with it, it cannot be empty. Override Module['canvas'] to specify
    // some other target to use, e.g. if the page contains multiple Unity
    // game instances.
    if (Module['canvas'] && !Module['canvas'].id) throw 'Module["canvas"] must
have a CSS ID associated with it!';
    var canvasId = Module['canvas'] ? Module['canvas'].id : 'unity-canvas';
    return '#' + jsDomCssEscapeId(canvasId);
}

function _JS_DOM_MapViewportCoordinateToElementLocalCoordinate(viewportX, viewportY,
targetX, targetY) {
    var canvas = document.querySelector(jsCanvasSelector());
    var rect = canvas && canvas.getBoundingClientRect();
    HEAPU32[targetX >> 2] = viewportX - (rect ? rect.left : 0);
    HEAPU32[targetY >> 2] = viewportY - (rect ? rect.top : 0);
}

function stringToNewUTF8(jsString) {
    var length = lengthBytesUTF8(jsString)+1;
    var cString = _malloc(length);
    stringToUTF8(jsString, cString, length);
    return cString;
}

function _JS_DOM_UnityCanvasSelector() {
    var canvasSelector = jsCanvasSelector();
    if (_JS_DOM_UnityCanvasSelector.selector != canvasSelector) {
        _free(_JS_DOM_UnityCanvasSelector.ptr);
        _JS_DOM_UnityCanvasSelector.ptr = stringToNewUTF8(canvasSelector);
        _JS_DOM_UnityCanvasSelector.selector = canvasSelector;
    }
    return _JS_DOM_UnityCanvasSelector.ptr;
}

function _JS_Eval_OpenURL(ptr)
{
    var str = UTF8ToString(ptr);
    window.open(str, '_blank', '');
}

var fs =
{numPendingSync:0, syncInternal:1000, syncInProgress:false, sync:function(onlyPendingSync)
{
    if (onlyPendingSync) {
        if (fs.numPendingSync == 0)
            return;
    }
    else if (fs.syncInProgress) {

```

```

        // this is to avoid indexedDB memory leak when FS.syncfs is executed
before the previous one completed.
        fs.numPendingSync++;
        return;
    }

    fs.syncInProgress = true;
    FS.syncfs(false, (function(err) {
        fs.syncInProgress = false;
    }));
    fs.numPendingSync = 0;
});
function _JS_FileSystem_Initialize()
{
    Module.setInterval(function() {
        fs.sync(true);
    }, fs.syncInternal);
}

function _JS_FileSystem_Sync()
{
    fs.sync(false);
}

var JS_GravitySensor = null;
function _JS_GravitySensor_IsRunning() {
    return (typeof GravitySensor !== 'undefined') ? (JS_GravitySensor &&
JS_GravitySensor.activated) : JS_GravitySensor_callback != 0;
}

function JS_GravitySensor_eventHandler() {
    if (JS_GravitySensor_callback != 0)
        dynCall_vfff(JS_GravitySensor_callback,
                    JS_GravitySensor.x * JS_Accelerometer_multiplier,
                    JS_GravitySensor.y * JS_Accelerometer_multiplier,
                    JS_GravitySensor.z * JS_Accelerometer_multiplier);
}

var JS_GravitySensor_frequencyRequest = 0;

var JS_LinearAccelerationSensor = null;

function JS_LinearAccelerationSensor_eventHandler() {
    var linearAccelerationValue = {
        x: JS_LinearAccelerationSensor.x * JS_Accelerometer_multiplier,
        y: JS_LinearAccelerationSensor.y * JS_Accelerometer_multiplier,
        z: JS_LinearAccelerationSensor.z * JS_Accelerometer_multiplier
    };
    if (JS_LinearAccelerationSensor_callback != 0)
        dynCall_vfff(JS_LinearAccelerationSensor_callback,
                    linearAccelerationValue.x, linearAccelerationValue.y, linearAccelerationValue.z);

        // Calculate and call the Gravity callback if the Gravity Sensor API isn't
present
    if (JS_GravitySensor_callback != 0 && typeof GravitySensor === 'undefined') {

```

```

        var gravityValue = JS_ComputeGravity(JS_Accelerometer_lastValue,
linearAccelerationValue);
            dynCall_vfff(JS_GravitySensor_callback, gravityValue.x, gravityValue.y,
gravityValue.z);
        }
    }

var JS_LinearAccelerationSensor_frequencyRequest = 0;

var JS_LinearAccelerationSensor_frequency = 0;
function JS_LinearAccelerationSensor_Start(callback, frequency) {
    // callback can be zero here when called via JS_GravitySensor_Start

    JS_DefineAccelerometerMultiplier();

    // If we don't have new sensor API, fallback to old DeviceMotionEvent
    if (typeof LinearAccelerationSensor === 'undefined') {
        JS_DeviceMotion_add(); // Must call before we set the callback
        if (callback != 0) JS_LinearAccelerationSensor_callback = callback;
        return;
    }

    if (callback != 0) JS_LinearAccelerationSensor_callback = callback;

    function InitializeLinearAccelerationSensor(frequency) {
        // Use device referenceFrame, since New Input System package does its own
compensation
        JS_LinearAccelerationSensor = new LinearAccelerationSensor({ frequency:
frequency, referenceFrame: 'device' });
        JS_LinearAccelerationSensor.addEventListener('reading',
JS_LinearAccelerationSensor_eventHandler);
        JS_LinearAccelerationSensor.addEventListener('error', function(e) {
            // e.error could be DOMException: Could not connect to a sensor
            warnOnce((e.error) ? e.error : e);
        });
        JS_LinearAccelerationSensor.start();
        JS_LinearAccelerationSensor_frequency = frequency;
    }

    // If the sensor is already created, stop and re-create it with new frequency
    if (JS_LinearAccelerationSensor) {
        if (JS_LinearAccelerationSensor_frequency != frequency) {
            JS_LinearAccelerationSensor.stop();
            JS_LinearAccelerationSensor.removeEventListener('reading',
JS_LinearAccelerationSensor_eventHandler);
            InitializeLinearAccelerationSensor(frequency);
        }
    }
    else if (JS_LinearAccelerationSensor_frequencyRequest != 0) {
        // If the permissions promise is currently in progress, then note new
frequency only
        JS_LinearAccelerationSensor_frequencyRequest = frequency;
    }
    else {
        JS_LinearAccelerationSensor_frequencyRequest = frequency;
    }
}

```

```

        // Request required permission for the LinearAccelerationSensor
        navigator.permissions.query({name: 'accelerometer'})
            .then(function(result) {
                if (result.state === "granted") {

InitializeLinearAccelerationSensor(JS_LinearAccelerationSensor_frequencyRequest);
                    } else {
                        warnOnce("No permission to use LinearAccelerationSensor.");
                    }
                    JS_LinearAccelerationSensor_frequencyRequest = 0;
                });
            }
        }

function _JS_GravitySensor_Start(callback, frequency) {
    assert(callback != 0, 'Invalid callback passed to JS_GravitySensor_Start');

        // If we don't have explicit new Gravity Sensor API, start the Accelerometer
        // and LinearAccelerationSensor
        // and we will compute the gravity value from those readings
        if (typeof GravitySensor === 'undefined') {
            // Start both Accelerometer and LinearAccelerationSensor
            _JS_Accelerometer_Start(0, Math.max(frequency,
JS_Accelerometer_frequency));
            _JS_LinearAccelerationSensor_Start(0, Math.max(frequency,
JS_LinearAccelerationSensor_frequency));

            // Add the gravity sensor callback (must be after Accelerometer and
            // LinearAccelerationSensor start)
            JS_GravitySensor_callback = callback;
            return;
        }

        JS_DefineAccelerometerMultiplier();

        JS_GravitySensor_callback = callback;

        function InitializeGravitySensor(frequency) {
            // Use device referenceFrame, since New Input System package does its own
            compensation
            JS_GravitySensor = new GravitySensor({ frequency: frequency,
referenceFrame: 'device' });
            JS_GravitySensor.addEventListener('reading',
JS_GravitySensor_eventHandler);
            JS_GravitySensor.addEventListener('error', function(e) {
                // e.error could be DOMException: Could not connect to a sensor
                warnOnce((e.error) ? e.error : e);
            });
            JS_GravitySensor.start();
        }

        // If the sensor is already created, stop and re-create it with new frequency
        if (JS_GravitySensor) {
            JS_GravitySensor.stop();
            JS_GravitySensor.removeEventListener('reading',
JS_GravitySensor_eventHandler);
    }
}

```

```

        InitializeGravitySensor(frequency);
    }
    else if (JS_GravitySensor_frequencyRequest != 0) {
        // If the permissions promise is currently in progress, then note new
frequency only
        JS_GravitySensor_frequencyRequest = frequency;
    }
    else {
        JS_GravitySensor_frequencyRequest = frequency;

        // Request required permission for the GravitySensor
        navigator.permissions.query({name: 'accelerometer'})
            .then(function(result) {
                if (result.state === "granted") {
                    InitializeGravitySensor(JS_GravitySensor_frequencyRequest);
                } else {
                    warnOnce("No permission to use GravitySensor.");
                }
                JS_GravitySensor_frequencyRequest = 0;
            });
    }
}

function JS_LinearAccelerationSensor_Stop() {
    if (!JS_LinearAccelerationSensor) {
        // Only actually stop the Linear Acceleration Sensor if we don't need it
to compute gravity values
        if (typeof GravitySensor !== 'undefined' || JS_GravitySensor_callback ==
0) {
            JS_LinearAccelerationSensor.stop();
            JS_LinearAccelerationSensor.removeEventListener('reading',
JS_LinearAccelerationSensor_eventHandler);
            JS_LinearAccelerationSensor = null;
        }
        JS_LinearAccelerationSensor_callback = 0;
        JS_LinearAccelerationSensor_frequency = 0;
    }
    else if (JS_LinearAccelerationSensor_callback != 0) {
        JS_LinearAccelerationSensor_callback = 0;
        JS_DeviceMotion_remove();
    }
}
function JS_GravitySensor_Stop() {
    JS_GravitySensor_callback = 0;

    // If we don't have Gravity Sensor API, stop the Accelerometer and
LinearAccelerationSensor
    if (typeof GravitySensor === 'undefined') {
        // Stop the source sensors if they're not used explicitly by Unity
        if (JS_Accelerometer_callback == 0) _JS_Accelerometer_Stop();
        if (JS_LinearAccelerationSensor_callback == 0)
_JS_LinearAccelerationSensor_Stop();
        return;
    }
}

```

```

        if (JS_GravitySensor) {
            JS_GravitySensor.stop();
            JS_GravitySensor.removeEventListener('reading',
JS_GravitySensor_eventHandler);
            JS_GravitySensor = null;
        }
    }

function _JS_GuardAgainstJsExceptions(cb) {
    try {
        (function() { dynCall_v.call(null, cb); })();
    } catch(e) {
        console.warn(e);
    }
}

var JS_Gyroscope = null;
function _JS_Gyroscope_IsRunning() {
    // Sensor is running if there is an activated new JS_Gyroscope; or the
JS_Gyroscope_callback is hooked up
    return (JS_Gyroscope && JS_Gyroscope.activated) || (JS_Gyroscope_callback != 0);
}
}

function JS_Gyroscope_eventHandler() {
    // Radians per second
    if (JS_Gyroscope_callback != 0)
        dynCall_vfff(JS_Gyroscope_callback, JS_Gyroscope.x, JS_Gyroscope.y,
JS_Gyroscope.z);
}

var JS_Gyroscope_frequencyRequest = 0;
function _JS_Gyroscope_Start(callback, frequency) {
    assert(callback != 0, 'Invalid callback passed to JS_Gyroscope_Start');

    // If we don't have new sensor API, fallback to old DeviceMotionEvent
    if (typeof Gyroscope === 'undefined') {
        JS_DeviceMotion_add(); // Must call before we set the callback
        JS_Gyroscope_callback = callback;
        return;
    }

    JS_Gyroscope_callback = callback;

    function InitializeGyroscope(frequency) {
        // Use device referenceFrame, since New Input System package does its own
compensation
        JS_Gyroscope = new Gyroscope({ frequency: frequency, referenceFrame:
'device' });
        JS_Gyroscope.addEventListener('reading', JS_Gyroscope_eventHandler);
        JS_Gyroscope.addEventListener('error', function(e) {
            // e.error could be DOMException: Could not connect to a sensor
            warnOnce((e.error) ? e.error : e);
        });
        JS_Gyroscope.start();
    }
}

```

```

        }

        // If the sensor is already created, stop and re-create it with new frequency
        if (JS_Gyroscope) {
            JS_Gyroscope.stop();
            JS_Gyroscope.removeEventListener('reading', JS_Gyroscope_eventHandler);
            InitializeGyroscope(frequency);
        }
        else if (JS_Gyroscope_frequencyRequest != 0) {
            // If the permissions promise is currently in progress, then note new
frequency only
            JS_Gyroscope_frequencyRequest = frequency;
        }
        else {
            JS_Gyroscope_frequencyRequest = frequency;

            // Request required permission for the Gyroscope
            navigator.permissions.query({name: 'gyroscope'})
                .then(function(result) {
                    if (result.state === "granted") {
                        InitializeGyroscope(JS_Gyroscope_frequencyRequest);
                    } else {
                        warnOnce("No permission to use Gyroscope.");
                    }
                    JS_Gyroscope_frequencyRequest = 0;
                });
        }
    }

function _JS_Gyroscope_Stop() {
    if (JS_Gyroscope) {
        JS_Gyroscope.stop();
        JS_Gyroscope.removeEventListener('reading', JS_Gyroscope_eventHandler);
        JS_Gyroscope = null;
        JS_Gyroscope_callback = 0;
    }
    else if (JS_Gyroscope_callback != 0) {
        JS_Gyroscope_callback = 0;
        JS_DeviceMotion_remove();
    }
}

function _JS_LinearAccelerationSensor_IsRunning() {
    // Sensor is running if there is an activated new
JS_LinearAccelerationSensor; or the JS_LinearAccelerationSensor_callback is hooked up
    return (JS_LinearAccelerationSensor && JS_LinearAccelerationSensor.activated)
|| (JS_LinearAccelerationSensor_callback != 0);
}

function _JS_Log_Dump(ptr, type)
{
    var str = UTF8ToString(ptr);
    if (typeof dump == 'function')

```

```

        dump (str);
    switch (type)
    {
        case 0: //LogType_Error
        case 1: //LogType_Assert
        case 4: //LogType_Exception
            console.error (str);
            return;

        case 2: //LogType_Warning
            console.warn (str);
            return;

        case 3: //LogType_Log
        case 5: //LogType_Debug
            console.log (str);
            return;

        default:
            console.error ("Unknown console message type!");
            console.error (str);
    }
}

function _JS_Log_StackTrace(buffer, bufferSize)
{
    var trace = stackTrace();
    if (buffer)
        stringToUTF8(trace, buffer, bufferSize);
    return lengthBytesUTF8(trace);
}

var mobile_input_hide_delay = null;
var mobile_input_text = null;
var mobile_input = null;
var mobile_input_ignore_blur_event = false;
function _JS_MobileKeyboard_GetIgnoreBlurEvent() {
    // On some platforms, such as iOS15, a blur event is sent to the window after the
    // keyboard
    // is closed. This causes the game to be paused in the blur event handler in
    // ScreenManagerWebGL.
    // It checks this return value to see if it should ignore the blur event.
    return mobile_input_ignore_blur_event;
}

function _JS_MobileKeyboard_GetKeyboardStatus()
{
    var kKeyboardStatusVisible = 0;
    var kKeyboardStatusDone = 1;
    //var kKeyboardStatusCanceled = 2;
    //var kKeyboardStatusLostFocus = 3;
    if (!mobile_input) return kKeyboardStatusDone;
    return kKeyboardStatusVisible;
}

```

```

function _JS_MobileKeyboard_GetText(buffer, bufferSize)
{
    // If the keyboard was closed, use the cached version of the input's text so that
Unity can
    // still ask for it.
    var text = mobile_input && mobile_input.input ? mobile_input.input.value :
        mobile_input_text ? mobile_input_text :
        "";
    if (buffer) stringToUTF8(text, buffer, bufferSize);
    return lengthBytesUTF8(text);
}

function _JS_MobileKeyboard_GetTextSelection(outStart, outLength)
{
    if (!mobile_input) {
        HEAP32[outStart >> 2] = 0;
        HEAP32[outLength >> 2] = 0;
        return;
    }
    HEAP32[outStart >> 2] = mobile_input.input.selectionStart;
    HEAP32[outLength >> 2] = mobile_input.input.selectionEnd -
mobile_input.input.selectionStart;
}

function _JS_MobileKeyboard_Hide(delay)
{
    if (mobile_input_hide_delay) return;
    mobile_input_ignore_blur_event = true;

    function hideMobileKeyboard() {
        if (mobile_input && mobile_input.input) {
            mobile_input_text = mobile_input.input.value;
            mobile_input.input = null;
            if (mobile_input.parentNode && mobile_input.parentNode) {
                mobile_input.parentNode.removeChild(mobile_input);
            }
        }
        mobile_input = null;
        mobile_input_hide_delay = null;

        // mobile_input_ignore_blur_event was set to true so that ScreenManagerWebGL
will ignore
        // the blur event it might get from the closing of the keyboard. But it might
not get that
        // blur event, too, depending on the browser. So we want to clear the flag,
as soon as we
        // can, but some time after the blur event has been potentially fired.
        setTimeout(function() {
            mobile_input_ignore_blur_event = false;
        }, 100);
    }

    if (delay) {
        // Delaying the hide of the input/keyboard allows a new input to be selected
and re-use the
    }
}

```

```

        // existing control. This fixes a problem where a quick tap select of a new
        // element would
        // cause it to not be displayed because it tried to be focused before the old
        // keyboard finished
        // sliding away.
        var hideDelay = 200;
        mobile_input_hide_delay = setTimeout(hideMobileKeyboard, hideDelay);
    } else {
        hideMobileKeyboard();
    }
}

function _JS_MobileKeyboard_SetCharacterLimit(limit)
{
    if (!mobile_input) return;
    mobile_input.input.maxLength = limit;
}

function _JS_MobileKeyboard_SetText(text)
{
    if (!mobile_input) return;
    text = UTF8ToString(text);
    mobile_input.input.value = text;
}

function _JS_MobileKeyboard_SetTextSelection(start, length)
{
    if (!mobile_input) return;
    mobile_input.input.setSelectionRange(start, start + length);
}

function _JS_MobileKeyboard_Show(text, keyboardType, autocorrection, multiline,
secure, alert,
                                placeholder, characterLimit)
{
    if (mobile_input_hide_delay) {
        clearTimeout(mobile_input_hide_delay);
        mobile_input_hide_delay = null;
    }

    text = UTF8ToString(text);
    mobile_input_text = text;

    placeholder = UTF8ToString(placeholder);

    var container = document.body;

    var hasExistingMobileInput = !!mobile_input;

    // From KeyboardOnScreen::KeyboardTypes
    var input_type;
    var KEYBOARD_TYPE_NUMBERS_AND_PUNCTUATION = 2;
    var KEYBOARD_TYPE_URL = 3;
    var KEYBOARD_TYPE_NUMBER_PAD = 4;
    var KEYBOARD_TYPE_PHONE_PAD = 5;
}

```

```

var KEYBOARD_TYPE_EMAIL_ADDRESS = 7;
if (!secure) {
    switch (keyboardType) {
        case KEYBOARD_TYPE_EMAIL_ADDRESS:
            input_type = "email";
            break;
        case KEYBOARD_TYPE_URL:
            input_type = "url";
            break;
        case KEYBOARD_TYPE_NUMBERS_AND_PUNCTUATION:
        case KEYBOARD_TYPE_NUMBER_PAD:
        case KEYBOARD_TYPE_PHONE_PAD:
            input_type = "number";
            break;
        default:
            input_type = "text";
            break;
    }
} else {
    input_type = "password";
}

if (hasExistingMobileInput) {
    if (mobile_input.multiline != multiline) {
        _JS_MobileKeyboard_Hide(false);
        return;
    }
}

var inputContainer = mobile_input || document.createElement("div");
if (!hasExistingMobileInput) {
    inputContainer.style = "width:100%; position:fixed; bottom:0px; margin:0px; padding:0px; left:0px; border: 1px solid #000; border-radius: 5px; background-color:#fff; font-size:14pt;";
    container.appendChild(inputContainer);
    mobile_input = inputContainer;
}

var input = hasExistingMobileInput ?
    mobile_input.input :
    document.createElement(multiline ? "textarea" : "input");

mobile_input.multiline = multiline;
mobile_input.secure = secure;
mobile_input.keyboardType = keyboardType;
mobile_input.inputType = input_type;

input.type = input_type;
input.style = "width:calc(100% - 85px); " + (multiline ? "height:100px;" : "") +
"vertical-align:top; border-radius: 5px; outline:none; cursor:default; resize:none; border:0px; padding:10px 0px 10px 10px;";

input.spellcheck = autocorrection ? true : false;
input.maxLength = characterLimit > 0 ? characterLimit : 524288;
input.value = text;

```

```

        input.placeholder = placeholder;

        if (!hasExistingMobileInput) {
            inputContainer.appendChild(input);
            inputContainer.input = input;
        }

        if (!hasExistingMobileInput) {
            var okButton = document.createElement("button");
            okButton.innerText = "OK";
            okButton.style = "border:0; position:absolute; left:calc(100% - 75px); top:0px; width:75px; height:100%; margin:0; padding:0; border-radius: 5px; background-color:#ffff";
            okButton.addEventListener("touchend", function() {
                _JS_MobileKeyboard_Hide(true);
            });

            inputContainer.appendChild(okButton);
            inputContainer.okButton = okButton;

            // For single-line text input, enter key will close the keyboard.
            input.addEventListener('keyup', function(e) {
                if (input.parentNode.multiline) return;
                if (e.code == 'Enter' || e.which == 13 || e.keyCode == 13) {
                    _JS_MobileKeyboard_Hide(true);
                }
            });
        }

        // On iOS, the keyboard has a done button that hides the keyboard. The only way to detect
        // when this happens seems to be when the HTML input loses focus, so we watch for the blur
        // event on the input element and close the element/keyboard when it's gotten.
        input.addEventListener("blur", function(e) {
            _JS_MobileKeyboard_Hide(true);
            e.stopPropagation();
            e.preventDefault();
        });

        input.select();
        input.focus();
    } else {
        input.select();
    }
}

var JS_OrientationSensor = null;

var JS_OrientationSensor_callback = 0;
function _JS_OrientationSensor_IsRunning() {
    // Sensor is running if there is an activated new JS_OrientationSensor; or the DeviceOrientation handler is hooked up
    return (JS_OrientationSensor && JS_OrientationSensor.activated) ||
(JS_OrientationSensor_callback != 0);
}

```

```

        }

    function JS_OrientationSensor_eventHandler() {
        if (JS_OrientationSensor_callback != 0)
            dynCall_vffff(JS_OrientationSensor_callback,
JS_OrientationSensor.quaternion[0], JS_OrientationSensor.quaternion[1],
JS_OrientationSensor.quaternion[2], JS_OrientationSensor.quaternion[3]);
    }

    var JS_OrientationSensor_frequencyRequest = 0;

    function JS_DeviceOrientation_eventHandler(event) {
        if (JS_OrientationSensor_callback) {
            // OBSERVATION: On Android Firefox, absolute = false,
webkitCompassHeading = null
            // OBSERVATION: On iOS Safari, absolute is undefined,
webkitCompassHeading and webkitCompassAccuracy are set

            // Convert alpha, beta, gamma Euler angles to a quaternion
            var degToRad = Math.PI / 180;
            var x = event.beta * degToRad;
            var y = event.gamma * degToRad;
            var z = event.alpha * degToRad;

            var cx = Math.cos(x/2);
            var sx = Math.sin(x/2);
            var cy = Math.cos(y/2);
            var sy = Math.sin(y/2);
            var cz = Math.cos(z/2);
            var sz = Math.sin(z/2);

            var qx = sx * cy * cz - cx * sy * sz;
            var qy = cx * sy * cz + sx * cy * sz;
            var qz = cx * cy * sz + sx * sy * cz;
            var qw = cx * cy * cz - sx * sy * sz;

            dynCall_vffff(JS_OrientationSensor_callback, qx, qy, qz, qw);
        }
    }

    function _JS_OrientationSensor_Start(callback, frequency) {
        assert(callback != 0, 'Invalid callback passed to
JS_OrientationSensor_Start');

        // If we don't have new sensor API, fallback to old DeviceOrientationEvent
        if (typeof RelativeOrientationSensor === 'undefined') {
            if (JS_OrientationSensor_callback == 0) {
                JS_OrientationSensor_callback = callback;
                JS_RequestDeviceSensorPermissions(1/*DeviceOrientationEvent
permission*/);
                window.addEventListener('deviceorientation',
JS_DeviceOrientation_eventHandler);
            }
            return;
        }
    }
}

```

```

JS_OrientationSensor_callback = callback;

function InitializeOrientationSensor(frequency) {
    // Use device referenceFrame, since New Input System package does its own
compensation
    // Use relative orientation to match native players
    JS_OrientationSensor = new RelativeOrientationSensor({ frequency:
frequency, referenceFrame: 'device' });
    JS_OrientationSensor.addEventListener('reading',
JS_OrientationSensor_eventHandler);
    JS_OrientationSensor.addEventListener('error', function(e) {
        // e.error could be DOMException: Could not connect to a sensor
        warnOnce((e.error) ? e.error : e);
    });
    JS_OrientationSensor.start();
}

// If the sensor is already created, stop and re-create it with new frequency
if (JS_OrientationSensor) {
    JS_OrientationSensor.stop();
    JS_OrientationSensor.removeEventListener('reading',
JS_OrientationSensor_eventHandler);
    InitializeOrientationSensor(frequency);
}
else if (JS_OrientationSensor_frequencyRequest != 0) {
    // If the permissions promise is currently in progress, then note new
frequency only
    JS_OrientationSensor_frequencyRequest = frequency;
}
else {
    JS_OrientationSensor_frequencyRequest = frequency;

    // Request required permissions for the RelativeOrientationSensor
    Promise.all([navigator.permissions.query({ name: "accelerometer" }),
                navigator.permissions.query({ name: "gyroscope" })])
        .then(function(results) {
            if (results.every(function(result) { return(result.state ===
"granted"); }))) {

InitializeOrientationSensor(JS_OrientationSensor_frequencyRequest);
            } else {
                warnOnce("No permissions to use RelativeOrientationSensor.");
            }
            JS_OrientationSensor_frequencyRequest = 0;
        });
}
}

function JS_OrientationSensor_Stop() {
    if (JS_OrientationSensor) {
        JS_OrientationSensor.stop();
        JS_OrientationSensor.removeEventListener('reading',
JS_OrientationSensor_eventHandler);
        JS_OrientationSensor = null;
    }
}

```

```

        else if (JS_OrientationSensor_callback != 0) {
            window.removeEventListener('deviceorientation',
JS_DeviceOrientation_eventHandler);
        }
        JS_OrientationSensor_callback = 0;
    }

function _JS_Profiler.InjectJobs()
{
    for (var jobname in Module["Jobs"])
    {
        var job = Module["Jobs"][jobname];
        if (typeof job["endtime"] != "undefined")
            Module.ccall("InjectProfilerSample", null, ["string", "number", "number"], [jobname, job.starttime, job.endtime]);
    }
}

function _JS_RequestDeviceSensorPermissionsOnTouch() {
    if (JS_DeviceSensorPermissions == 0) return;

    // Re-request any required device sensor permissions (iOS requires that
permissions are requested on a user interaction event)
    JS_RequestDeviceSensorPermissions(JS_DeviceSensorPermissions);
}

function _JS_RunQuitCallbacks() {
    Module.QuitCleanup();
}

var JS_ScreenOrientation_callback = 0;
function JS_ScreenOrientation_eventHandler() {
    if (JS_ScreenOrientation_callback)
        dynCall_viii(JS_ScreenOrientation_callback, window.innerWidth, window.innerHeight,
screen.orientation ? screen.orientation.angle : window.orientation);
}
function _JS_ScreenOrientation_DeInit() {
    JS_ScreenOrientation_callback = 0;
    window.removeEventListener('resize', JS_ScreenOrientation_eventHandler);
    if (screen.orientation)
        screen.orientation.removeEventListener('change',
JS_ScreenOrientation_eventHandler);
}
function _JS_ScreenOrientation_Init(callback) {
    // Only register if not yet registered
    if (!JS_ScreenOrientation_callback) {
        if (screen.orientation) {
            // Use Screen Orientation API if available:
            // - https://www.w3.org/TR/screen-orientation/
            // - https://caniuse.com/screen-orientation
            // - https://developer.mozilla.org/en-
US/docs/Web/API/Screen/orientation
            // (Firefox, Chrome, Chrome for Android, Firefox for Android)
            screen.orientation.addEventListener('change',
JS_ScreenOrientation_eventHandler);
    }
}

```

```

        }

        // As a fallback, use deprecated DOM window.orientation field if
available:
        // - https://compat.spec.whatwg.org/#dom-window-orientation
        // - https://developer.mozilla.org/en-US/docs/Web/API/Window/orientation
        // (Safari for iOS)
        // Listening to resize event also helps emulate landscape/portrait
transitions on desktop
        // browsers when the browser window is scaled to narrow/wide
configurations.
        window.addEventListener('resize', JS_ScreenOrientation_eventHandler);

        JS_ScreenOrientation_callback = callback;

        // Trigger the event handler immediately after the engine
initialization is done to start up
        // ScreenManager with the initial state.
        setTimeout(JS_ScreenOrientation_eventHandler, 0);
    }
}

var JS_ScreenOrientation_requestedLockType = -1;

var JS_ScreenOrientation_appliedLockType = -1;

var JS_ScreenOrientation_timeoutID = -1;
function _JS_ScreenOrientation_Lock(orientationLockType) {
    // We will use the Screen Orientation API if available, and silently return
if not available
    // - https://www.w3.org/TR/screen-orientation/
    // - https://caniuse.com/screen-orientation
    // - https://developer.mozilla.org/en-US/docs/Web/API/Screen/orientation
    if (!screen.orientation) {
        // As of writing, this is only not implemented on Safari
        return;
    }

    // Callback to apply the lock
    function applyLock() {
        JS_ScreenOrientation_appliedLockType =
JS_ScreenOrientation_requestedLockType;

        // Index must match enum class OrientationLockType in
ScreenOrientation.h
        var screenOrientations = ['any', 0/*natural*/, 'landscape',
'portrait', 'portrait-primary', 'portrait-secondary', 'landscape-primary', 'landscape-
secondary' ];
        var type = screenOrientations[JS_ScreenOrientation_appliedLockType];

        assert(type, 'Invalid orientationLockType passed to
JS_ScreenOrientation_Lock');

        // Apply the lock, which is done asynchronously and returns a Promise
        screen.orientation.lock(type).then(function() {
            // Upon success, see if the
JS_ScreenOrientation_requestedLockType value has changed, in which case, we will now
need to queue another applyLock

```

```
    if (JS_ScreenOrientation_requestedLockType !=  
JS_ScreenOrientation_appliedLockType) {  
        JS_ScreenOrientation_timeoutID = setTimeout(applyLock,  
0);  
    }  
}
```

```

        else {
            JS_ScreenOrientation_timeoutID = -1;
        }
    }).catch(function(err) {
        // When screen.orientation.lock() is called on a desktop
        // browser, a DOMException is thrown by the promise
        warnOnce(err);
        JS_ScreenOrientation_timeoutID = -1;
    });

        // Note, there is also an screen.orientation.unlock() which unlocks
        auto rotate to default orientation.
        // On my Google Pixel 5, this allows 'portrait-primary' AND
        'landscape', but will differ depending on device.
    }

    // Request this orientationLockType be applied on the callback
    JS_ScreenOrientation_requestedLockType = orientationLockType;

    // Queue applyLock callback if there is not already a callback or a
    // screen.orientation.lock call in progress
    if (JS_ScreenOrientation_timeoutID == -1 && orientationLockType !=
    JS_ScreenOrientation_appliedLockType) {
        JS_ScreenOrientation_timeoutID = setTimeout(applyLock, 0);
    }
}

var WEBAudio =
{audioInstanceIdCounter:0,audioInstances:{},audioContext:null, audioWebEnabled:0, audioCa
che:[], pendingAudioSources:{}};
function _JS_Sound_ResumeIfNeeded()
{
    if (WEBAudio.audioWebEnabled == 0)
        return;

    if (WEBAudio.audioContext.state === 'suspended')
        WEBAudio.audioContext.resume().catch(function (error) {
            console.warn("Could not resume audio context. Exception: " + error);
        });
}

function _JS_SystemInfo_GetBrowserName(buffer, bufferSize)
{
    var browser = Module.SystemInfo.browser;
    if (buffer)
        stringToUTF8(browser, buffer, bufferSize);
    return lengthBytesUTF8(browser);
}

function _JS_SystemInfo_GetBrowserVersionString(buffer, bufferSize)
{
    var browserVer = Module.SystemInfo.browserVersion;
    if (buffer)
        stringToUTF8(browserVer, buffer, bufferSize);
    return lengthBytesUTF8(browserVer);
}

```

```

}

function _JS_SystemInfo_GetCanvasClientSize(domElementSelector, outWidth, outHeight)
{
    var selector = UTF8ToString(domElementSelector);
    var canvas = (selector == '#canvas') ? Module['canvas'] :
document.querySelector(selector);
    var w = 0, h = 0;
    if (canvas) {
        var size = canvas.getBoundingClientRect();
        w = size.width;
        h = size.height;
    }
    HEAPF64[outWidth >> 3] = w;
    HEAPF64[outHeight >> 3] = h;
}

function _JS_SystemInfo_GetDocumentURL(buffer, bufferSize)
{
    if (buffer)
        stringToUTF8(document.URL, buffer, bufferSize);
    return lengthBytesUTF8(document.URL);
}

function _JS_SystemInfo_GetGPUInfo(buffer, bufferSize)
{
    var gpuinfo = Module.SystemInfo.gpu;
    if (buffer)
        stringToUTF8(gpuinfo, buffer, bufferSize);
    return lengthBytesUTF8(gpuinfo);
}

function _JS_SystemInfo_GetLanguage(buffer, bufferSize)
{
    var language = Module.SystemInfo.language;
    if (buffer)
        stringToUTF8(language, buffer, bufferSize);
    return lengthBytesUTF8(language);
}

function _JS_SystemInfo_GetMatchWebGLToCanvasSize()
{
    // If matchWebGLToCanvasSize is not present, it is
    // same as true, to keep backwards compatibility with user page templates
    // that are not setting this field.
    return Module.matchWebGLToCanvasSize || Module.matchWebGLToCanvasSize ===
undefined;
}

function _JS_SystemInfo_GetMemory()
{
    return HEAPU8.length/(1024*1024);
}

function _JS_SystemInfo_GetOS(buffer, bufferSize)
```

```

    {
        var browser = Module.SystemInfo.os + " " + Module.SystemInfo.osVersion;
        if (buffer)
            stringToUTF8(browser, buffer, bufferSize);
        return lengthBytesUTF8(browser);
    }

function _JS_SystemInfo_GetPreferredDevicePixelRatio()
{
    return Module.matchWebGLToCanvasSize == false ? 1 : Module.devicePixelRatio
|| window.devicePixelRatio || 1;
}

function _JS_SystemInfo_GetScreenSize(outWidth, outHeight)
{
    HEAPF64[outWidth >> 3] = Module.SystemInfo.width;
    HEAPF64[outHeight >> 3] = Module.SystemInfo.height;
}

function _JS_SystemInfo_HasAstcHdr()
{
    var ext = GLctx.getExtension('WEBGL_compressed_texture_astc');
    if (ext && ext.getSupportedProfiles) {
        return ext.getSupportedProfiles().includes("hdr");
    }
    return false;
}

function _JS_SystemInfo_HasCursorLock()
{
    return Module.SystemInfo.hasCursorLock;
}

function _JS_SystemInfo_HasFullscreen()
{
    return Module.SystemInfo.hasFullscreen;
}

function _JS_SystemInfo_HasWebGL()
{
    return Module.SystemInfo.hasWebGL;
}

function _JS_UnityEngineShouldQuit() {
    return !Module.shouldQuit;
}

var wr = {requests:{},responses:{},abortControllers:{},timer:{},nextRequestId:1};
function _JS_WebRequest_Abort(requestId)
{
    var abortController = wr.abortControllers[requestId];
    if (!abortController || abortController.signal.aborted) {
        return;
    }
}

```

```

        abortController.abort();
    }

function _JS_WebRequest_Create(url, method)
{
    var _url = UTF8ToString(url);
    var _method = UTF8ToString(method);
    var abortController = new AbortController();
    var requestOptions = {
        url: _url,
        init: {
            method: _method,
            signal: abortController.signal,
            headers: {},
            enableStreamingDownload: true
        },
        tempBuffer: null,
        tempBufferSize: 0
    };

    wr.abortControllers[wr.nextRequestId] = abortController;
    wr.requests[wr.nextRequestId] = requestOptions;

    return wr.nextRequestId++;
}

function jsWebRequestGetResponseHeaderString(requestId) {
    var response = wr.responses[requestId];
    if (!response) {
        return "";
    }

    // Use cached value of response header string if present
    if (response.headerString) {
        return response.headerString;
    }

    // Create response header string from headers object
    var headers = "";
    var entries = response.headers.entries();
    for (var result = entries.next(); !result.done; result = entries.next()) {
        headers += result.value[0] + ": " + result.value[1] + "\r\n";
    }
    response.headerString = headers;

    return headers;
}
function _JS_WebRequest_GetResponseMetaData(requestId, headerBuffer, headerSize,
responseUrlBuffer, responseUrlSize)
{
    var response = wr.responses[requestId];
    if (!response) {
        stringToUTF8("", headerBuffer, headerSize);
        stringToUTF8("", responseUrlBuffer, responseUrlSize);
        return;
    }
}
```

```

    }

    if (headerBuffer) {
        var headers = jsWebRequestGetResponseHeaderString(requestId);
        stringToUTF8(headers, headerBuffer, headerSize);
    }

    if (responseUrlBuffer) {
        stringToUTF8(response.url, responseUrlBuffer, responseUrlSize);
    }
}

function _JS_WebRequest_GetResponseMetaLengths(requestId, buffer)
{
    var response = wr.responses[requestId];
    if (!response) {
        HEAPU32[buffer >> 2] = 0;
        HEAPU32[(buffer >> 2) + 1] = 0;
        return;
    }

    var headers = jsWebRequestGetResponseHeaderString(requestId);

    // Set length of header and response url to output buffer
    HEAPU32[buffer >> 2] = lengthBytesUTF8(headers);
    HEAPU32[(buffer >> 2) + 1] = lengthBytesUTF8(response.url);
}

function _JS_WebRequest_Release(requestId)
{
    // Clear timeout
    if (wr.timer[requestId]) {
        clearTimeout(wr.timer[requestId]);
    }

    // Remove all resources for request
    delete wr.requests[requestId];
    delete wr.responses[requestId];
    delete wr.abortControllers[requestId];
    delete wr.timer[requestId];
}

function _JS_WebRequest_Send(requestId, ptr, length, arg, onresponse, onprogress)
{
    var requestOptions = wr.requests[requestId];
    var abortController = wr.abortControllers[requestId];

    function getTempBuffer(size) {
        // Allocate new temp buffer if none has been allocated
        if (!requestOptions.tempBuffer) {
            const initialSize = Math.max(size, 1024); // Use 1 kB as
minimal temp buffer size to prevent too many reallocations
            requestOptions.tempBuffer = _malloc(initialSize);
            requestOptions.tempBufferSize = initialSize;
        }
    }
}

```

```

        // Increase size of temp buffer if necessary
        if (requestOptions.tempBufferSize < size) {
            _free(requestOptions.tempBuffer);
            requestOptions.tempBuffer = _malloc(size);
            requestOptions.tempBufferSize = size;
        }

        return requestOptions.tempBuffer;
    }

    function ClearTimeout() {
        if (wr.timer[requestId]) {
            clearTimeout(wr.timer[requestId]);
            delete wr.timer[requestId];
        }
    }

    function HandleSuccess(response, body) {
        ClearTimeout();

        if (!onresponse) {
            return;
        }

        var kWebRequestOK = 0;
        // 200 is successful http request, 0 is returned by non-http requests
(file:).

        if (requestOptions.init.enableStreamingDownload) {
            // Body was streamed only send final body length
            dynCall('viiiiii', onresponse, [arg, response.status, 0,
body.length, 0, kWebRequestOK]);
        } else if (body.length != 0) {
            // Send whole body at once
            var buffer = _malloc(body.length);
            HEAPU8.set(body, buffer);
            dynCall('viiiiii', onresponse, [arg, response.status, buffer,
body.length, 0, kWebRequestOK]);
        } else {
            dynCall('viiiiii', onresponse, [arg, response.status, 0, 0, 0,
kWebRequestOK]);
        }

        // Cleanup temp buffer
        if (requestOptions.tempBuffer) {
            _free(requestOptions.tempBuffer);
        }
    }

    function HandleError(err, code) {
        ClearTimeout();

        if (!onresponse) {
            return;
        }

        var len = lengthBytesUTF8(err) + 1;

```

```

        var buffer = _malloc(len);
        stringToUTF8(err, buffer, len);
        dynCall('viiiiii', onresponse, [arg, 500, 0, 0, buffer, code]);
        _free(buffer);

        // Clean up temp buffer
        if (requestOptions.tempBuffer) {
            _free(requestOptions.tempBuffer);
        }
    }

    function HandleProgress(e) {
        if (!onprogress || !e.lengthComputable) {
            return;
        }

        var response = e.response;
        wr.responses[requestId] = response;

        if (e.chunk) {
            // Response body streaming is enabled copy data to new buffer
            var buffer = getTempBuffer(e.chunk.length);
            HEAPU8.set(e.chunk, buffer);
            dynCall('viiiiiii', onprogress, [arg, response.status, e.loaded,
e.total, buffer, e.chunk.length]);
        } else {
            // no response body streaming
            dynCall('viiiiiii', onprogress, [arg, response.status, e.loaded,
e.total, 0, 0]);
        }
    }

    try {
        if (length > 0) {
            var postData = HEAPU8.subarray(ptr, ptr+length);
            requestOptions.init.body = new Blob([postData]);
        }

        // Add timeout handler if timeout is set
        if (requestOptions.timeout) {
            wr.timer[requestId] = setTimeout(function () {
                requestOptions.isTimedOut = true;
                abortController.abort();
            }, requestOptions.timeout);
        }

        var fetchImpl = Module.fetchWithProgress;
        requestOptions.init.onProgress = HandleProgress;
        if (Module.companyName && Module.productName && Module.cachedFetch) {
            fetchImpl = Module.cachedFetch;
            requestOptions.init.companyName = Module.companyName;
            requestOptions.init.productName = Module.productName;
            requestOptions.init.productVersion = Module.productVersion;
            requestOptions.init.control =
Module.cacheControl(requestOptions.url);
        }
    }
}

```

```

        fetchImpl(requestOptions.url, requestOptions.init).then(function
(response) {
            wr.responses[requestId] = response;

            HandleSuccess(response, response.parsedBody);
        }).catch(function (error) {
            var kWebErrorUnknown = 2;
            var kWebErrorAborted = 17;
            var kWebErrorTimeout = 14;

            if (requestOptions.isTimedOut) {
                HandleError("Connection timed out.", kWebErrorTimeout);
            } else if (abortController.signal.aborted) {
                HandleError("Aborted.", kWebErrorAborted);
            } else {
                HandleError(error.message, kWebErrorUnknown);
            }
        });
    } catch(error) {
        var kWebErrorUnknown = 2;
        HandleError(error.message, kWebErrorUnknown);
    }
}

function _JS_WebRequest_SetRedirectLimit(request, redirectLimit)
{
    var requestOptions = wr.requests[request];
    if (!requestOptions) {
        return;
    }

    // Disable redirects if redirectLimit == 0 otherwise use browser defined
    redirect limit
    requestOptions.init.redirect = redirectLimit === 0 ? "error" : "follow";
}

function _JS_WebRequest_SetRequestHeader(requestId, header, value)
{
    var requestOptions = wr.requests[requestId];
    if (!requestOptions) {
        return;
    }

    var _header = UTF8ToString(header);
    var _value = UTF8ToString(value);
    requestOptions.init.headers[_header] = _value;
}

function _JS_WebRequest_SetTimeout(requestId, timeout)
{
    var requestOptions = wr.requests[requestId];
    if (!requestOptions) {
        return;
    }

    requestOptions.timeout = timeout;
}

```

```
}

function __assert_fail(condition, filename, line, func) {
    abort('Assertion failed: ' + UTF8ToString(condition) + ', at: ' + [filename ?
    UTF8ToString(filename) : 'unknown filename', line, func ? UTF8ToString(func) : 'unknown
    function']);
}

function __cxa_allocate_exception(size) {
    // Thrown object is prepended by exception metadata block
    return _malloc(size + 16) + 16;
}

/** @constructor */
function ExceptionInfo(excPtr) {
    this.excPtr = excPtr;
    this.ptr = excPtr - 16;

    this.set_type = function(type) {
        HEAP32[((this.ptr)+(4))>>2] = type;
    };

    this.get_type = function() {
        return HEAP32[((this.ptr)+(4))>>2];
    };

    this.set_destructor = function(destructor) {
        HEAP32[((this.ptr)+(8))>>2] = destructor;
    };

    this.get_destructor = function() {
        return HEAP32[((this.ptr)+(8))>>2];
    };

    this.set_refcount = function(refcount) {
        HEAP32[((this.ptr)>>2)] = refcount;
    };

    this.set_caught = function (caught) {
        caught = caught ? 1 : 0;
        HEAP8[((this.ptr)+(12))>>0] = caught;
    };

    this.get_caught = function () {
        return HEAP8[((this.ptr)+(12))>>0] != 0;
    };

    this.set_rethrown = function (rethrown) {
        rethrown = rethrown ? 1 : 0;
        HEAP8[((this.ptr)+(13))>>0] = rethrown;
    };

    this.get_rethrown = function () {
        return HEAP8[((this.ptr)+(13))>>0] != 0;
    };
}
```

```

// Initialize native structure fields. Should be called once after allocated.
this.init = function(type, destructor) {
    this.set_type(type);
    this.set_destructor(destructor);
    this.set_refcount(0);
    this.set_caught(false);
    this.set_rethrown(false);
}

this.add_ref = function() {
    var value = HEAP32[((this.ptr)>>2)];
    HEAP32[((this.ptr)>>2)] = value + 1;
};

// Returns true if last reference released.
this.release_ref = function() {
    var prev = HEAP32[((this.ptr)>>2)];
    HEAP32[((this.ptr)>>2)] = prev - 1;
    assert(prev > 0);
    return prev === 1;
};
}

/**
 * @constructor
 * @param {number=} ptr
 */
function CatchInfo(ptr) {

    this.free = function() {
        _free(this.ptr);
        this.ptr = 0;
    };

    this.set_base_ptr = function(basePtr) {
        HEAP32[((this.ptr)>>2)] = basePtr;
    };

    this.get_base_ptr = function() {
        return HEAP32[((this.ptr)>>2)];
    };

    this.set_adjusted_ptr = function(adjustedPtr) {
        HEAP32[((this.ptr)+(4))>>2] = adjustedPtr;
    };

    this.get_adjusted_ptr_addr = function() {
        return this.ptr + 4;
    }

    this.get_adjusted_ptr = function() {
        return HEAP32[((this.ptr)+(4))>>2];
    };
}

```

```

// Get pointer which is expected to be received by catch clause in C++ code. It
may be adjusted
// when the pointer is casted to some of the exception object base classes (e.g.
when virtual
// inheritance is used). When a pointer is thrown this method should return the
thrown pointer
// itself.
this.get_exception_ptr = function() {
    // Work around a fastcomp bug, this code is still included for some reason in a
build without
    // exceptions support.
    var isPointer = __cxa_is_pointer_type(
        this.get_exception_info().get_type());
    if (isPointer) {
        return HEAP32[((this.get_base_ptr())>>2)];
    }
    var adjusted = this.get_adjusted_ptr();
    if (adjusted !== 0) return adjusted;
    return this.get_base_ptr();
};

this.get_exception_info = function() {
    return new ExceptionInfo(this.get_base_ptr());
};

if (ptr === undefined) {
    this.ptr = _malloc(8);
    this.set_adjusted_ptr(0);
} else {
    this.ptr = ptr;
}
}

var exceptionCaught = [];

function exception_addRef(info) {
    info.add_ref();
}

var uncaughtExceptionCount = 0;
function __cxa_begin_catch(ptr) {
    var catchInfo = new CatchInfo(ptr);
    var info = catchInfo.get_exception_info();
    if (!info.get_caught()) {
        info.set_caught(true);
        uncaughtExceptionCount--;
    }
    info.set_rethrown(false);
    exceptionCaught.push(catchInfo);
    exception_addRef(info);
    return catchInfo.get_exception_ptr();
}

var exceptionLast = 0;

```

```

function __cxa_free_exception(ptr) {
    try {
        return _free(new ExceptionInfo(ptr).ptr);
    } catch(e) {
        err('exception during cxa_free_exception: ' + e);
    }
}

function exception_decRef(info) {
    // A rethrown exception can reach refcount 0; it must not be discarded
    // Its next handler will clear the rethrown flag and addRef it, prior to
    // final decRef and destruction here
    if (info.release_ref() && !info.get_rethrown()) {
        var destructor = info.get_destructor();
        if (destructor) {
            // In Wasm, destructors return 'this' as in ARM
            (function(a1) { return dynCall_ii.apply(null, [destructor, a1]));
        })(info.excPtr);
    }
    __cxa_free_exception(info.excPtr);
}

function __cxa_end_catch() {
    // Clear state flag.
    _setThrew(0);
    assert(exceptionCaught.length > 0);
    // Call destructor if one is registered then clear it.
    var catchInfo = exceptionCaught.pop();

    exception_decRef(catchInfo.get_exception_info());
    catchInfo.free();
    exceptionLast = 0; // XXX in decRef?
}

function __resumeException(catchInfoPtr) {
    var catchInfo = new CatchInfo(catchInfoPtr);
    var ptr = catchInfo.get_base_ptr();
    if (!exceptionLast) { exceptionLast = ptr; }
    catchInfo.free();
    throw ptr;
}

function __cxa_find_matching_catch_2() {
    var thrown = exceptionLast;
    if (!thrown) {
        // just pass through the null ptr
        setTempRet0(0); return ((0)|0);
    }
    var info = new ExceptionInfo(thrown);
    var thrownType = info.get_type();
    var catchInfo = new CatchInfo();
    catchInfo.set_base_ptr(thrown);
    catchInfo.set_adjusted_ptr(thrown);
    if (!thrownType) {
        // just pass through the thrown ptr
        setTempRet0(0); return ((catchInfo.ptr)|0);
    }
}

```

```

var typeArray = Array.prototype.slice.call(arguments);

// canCatch receives a **, add indirection
// The different catch blocks are denoted by different types.
// Due to inheritance, those types may not precisely match the
// type of the thrown object. Find one which matches, and
// return the type of the catch block which should be called.
for (var i = 0; i < typeArray.length; i++) {
    var caughtType = typeArray[i];
    if (caughtType === 0 || caughtType === thrownType) {
        // Catch all clause matched or exactly the same type is caught
        break;
    }
    if (__cxa_can_catch(caughtType, thrownType,
catchInfo.get_adjusted_ptr_addr())) {
        setTempRet0(caughtType); return ((catchInfo.ptr)|0);
    }
}
setTempRet0(thrownType); return ((catchInfo.ptr)|0);
}

function __cxa_find_matching_catch_3() {
    var thrown = exceptionLast;
    if (!thrown) {
        // just pass through the null ptr
        setTempRet0(0); return ((0)|0);
    }
    var info = new ExceptionInfo(thrown);
    var thrownType = info.get_type();
    var catchInfo = new CatchInfo();
    catchInfo.set_base_ptr(thrown);
    catchInfo.set_adjusted_ptr(thrown);
    if (!thrownType) {
        // just pass through the thrown ptr
        setTempRet0(0); return ((catchInfo.ptr)|0);
    }
    var typeArray = Array.prototype.slice.call(arguments);

    // canCatch receives a **, add indirection
    // The different catch blocks are denoted by different types.
    // Due to inheritance, those types may not precisely match the
    // type of the thrown object. Find one which matches, and
    // return the type of the catch block which should be called.
    for (var i = 0; i < typeArray.length; i++) {
        var caughtType = typeArray[i];
        if (caughtType === 0 || caughtType === thrownType) {
            // Catch all clause matched or exactly the same type is caught
            break;
        }
        if (__cxa_can_catch(caughtType, thrownType,
catchInfo.get_adjusted_ptr_addr())) {
            setTempRet0(caughtType); return ((catchInfo.ptr)|0);
        }
    }
    setTempRet0(thrownType); return ((catchInfo.ptr)|0);
}

```

```

}

function __cxa_find_matching_catch_4() {
    var thrown = exceptionLast;
    if (!thrown) {
        // just pass through the null ptr
        setTempRet0(0); return ((0)|0);
    }
    var info = new ExceptionInfo(thrown);
    var thrownType = info.get_type();
    var catchInfo = new CatchInfo();
    catchInfo.set_base_ptr(thrown);
    catchInfo.set_adjusted_ptr(thrown);
    if (!thrownType) {
        // just pass through the thrown ptr
        setTempRet0(0); return ((catchInfo.ptr)|0);
    }
    var typeArray = Array.prototype.slice.call(arguments);

    // canCatch receives a **, add indirection
    // The different catch blocks are denoted by different types.
    // Due to inheritance, those types may not precisely match the
    // type of the thrown object. Find one which matches, and
    // return the type of the catch block which should be called.
    for (var i = 0; i < typeArray.length; i++) {
        var caughtType = typeArray[i];
        if (caughtType === 0 || caughtType === thrownType) {
            // Catch all clause matched or exactly the same type is caught
            break;
        }
        if (__cxa_can_catch(caughtType, thrownType,
catchInfo.get_adjusted_ptr_addr())) {
            setTempRet0(caughtType); return ((catchInfo.ptr)|0);
        }
    }
    setTempRet0(thrownType); return ((catchInfo.ptr)|0);
}

function __cxa_rethrow() {
    var catchInfo = exceptionCaught.pop();
    if (!catchInfo) {
        abort('no exception to throw');
    }
    var info = catchInfo.get_exception_info();
    var ptr = catchInfo.get_base_ptr();
    if (!info.get_rethrown()) {
        // Only pop if the corresponding push was through rethrow_primary_exception
        exceptionCaught.push(catchInfo);
        info.set_rethrown(true);
        info.set_caught(false);
        uncaughtExceptionCount++;
    } else {
        catchInfo.free();
    }
}

```

```

exceptionLast = ptr;
throw ptr;
}

function __cxa_throw(ptr, type, destructor) {
    var info = new ExceptionInfo(ptr);
    // Initialize ExceptionInfo content after it was allocated in
__cxa_allocate_exception.
    info.init(type, destructor);
    exceptionLast = ptr;
    uncaughtExceptionCount++;
    throw ptr;
}

var PATH = {splitPath:function(filename) {
    var splitPathRe =
/^((?:[^?]+|(?:(?:\.\.){1,2}|[^\/]+|\\.\\.[^\/]*|))|(.?:[\/]*$))/;
    return splitPathRe.exec(filename).slice(1);
},normalizeArray:function(parts, allowAboveRoot) {
    // if the path tries to go above the root, `up` ends up > 0
    var up = 0;
    for (var i = parts.length - 1; i >= 0; i--) {
        var last = parts[i];
        if (last === '.') {
            parts.splice(i, 1);
        } else if (last === '..') {
            parts.splice(i, 1);
            up++;
        } else if (up) {
            parts.splice(i, 1);
            up--;
        }
    }
    // if the path is allowed to go above the root, restore leading ..
    if (allowAboveRoot) {
        for (; up; up--) {
            parts.unshift('..');
        }
    }
    return parts;
},normalize:function(path) {
    var isAbsolute = path.charAt(0) === '/',
        trailingSlash = path.substr(-1) === '/';
    // Normalize the path
    path = PATH.normalizeArray(path.split('/').filter(function(p) {
        return !p;
    }), !isAbsolute).join('/');
    if (!path && !isAbsolute) {
        path = '.';
    }
    if (path && trailingSlash) {
        path += '/';
    }
    return (isAbsolute ? '/' : '') + path;
}
}

```

```

}, dirname: function(path) {
  var result = PATH.splitPath(path),
    root = result[0],
    dir = result[1];
  if (!root && !dir) {
    // No dirname whatsoever
    return '.';
  }
  if (dir) {
    // It has a dirname, strip trailing slash
    dir = dir.substr(0, dir.length - 1);
  }
  return root + dir;
}, basename: function(path) {
  // EMSCRIPтен return '' for '/', not an empty string
  if (path === '/') return '/';
  path = PATH.normalize(path);
  path = path.replace(/\//g, "");
  var lastSlash = path.lastIndexOf('/');
  if (lastSlash === -1) return path;
  return path.substr(lastSlash+1);
}, extname: function(path) {
  return PATH.splitPath(path)[3];
}, join: function() {
  var paths = Array.prototype.slice.call(arguments, 0);
  return PATH.normalize(paths.join('/'));
}, join2: function(l, r) {
  return PATH.normalize(l + '/' + r);
}};

function getRandomDevice() {
  if (typeof crypto == 'object' && typeof crypto['getRandomValues'] == 'function')
{
    // for modern web browsers
    var randomBuffer = new Uint8Array(1);
    return function() { crypto.getRandomValues(randomBuffer); return
randomBuffer[0]; };
  } else
  if (ENVIRONMENT_IS_NODE) {
    // for nodejs with or without crypto support included
    try {
      var crypto_module = require('crypto');
      // nodejs has crypto support
      return function() { return crypto_module['randomBytes'](1)[0]; };
    } catch (e) {
      // nodejs doesn't have crypto support
    }
  }
  // we couldn't find a proper implementation, as Math.random() is not suitable for
  // /dev/random, see emscripten-core/emscripten/pull/7096
  return function() { abort("no cryptographic support found for randomDevice.
consider polyfilling it if you want to use something insecure like Math.random(), e.g.
put this in a --pre-js: var crypto = { getRandomValues: function(array) { for (var i =
0; i < array.length; i++) array[i] = (Math.random()*256)|0 } }; ); };
}

```

```

var PATH_FS = {resolve:function() {
  var resolvedPath = '',
    resolvedAbsolute = false;
  for (var i = arguments.length - 1; i >= -1 && !resolvedAbsolute; i--) {
    var path = (i >= 0) ? arguments[i] : FS.cwd();
    // Skip empty and invalid entries
    if (typeof path != 'string') {
      throw new TypeError('Arguments to path.resolve must be strings');
    } else if (!path) {
      return '';
    }
    resolvedPath = path + '/' + resolvedPath;
    resolvedAbsolute = path.charAt(0) === '/';
  }
  // At this point the path should be resolved to a full absolute path, but
  // handle relative paths to be safe (might happen when process.cwd() fails)
  resolvedPath = PATH.normalizeArray(resolvedPath.split('/').filter(function(p) {
    return !!p;
  }), !resolvedAbsolute).join('/');
  return ((resolvedAbsolute ? '/' : '') + resolvedPath) || '.';
},relative:function(from, to) {
  from = PATH_FS.resolve(from).substr(1);
  to = PATH_FS.resolve(to).substr(1);
  function trim(arr) {
    var start = 0;
    for (; start < arr.length; start++) {
      if (arr[start] !== '') break;
    }
    var end = arr.length - 1;
    for (; end >= 0; end--) {
      if (arr[end] !== '') break;
    }
    if (start > end) return [];
    return arr.slice(start, end - start + 1);
  }
  var fromParts = trim(from.split('/'));
  var toParts = trim(to.split('/'));
  var length = Math.min(fromParts.length, toParts.length);
  var samePartsLength = length;
  for (var i = 0; i < length; i++) {
    if (fromParts[i] !== toParts[i]) {
      samePartsLength = i;
      break;
    }
  }
  var outputParts = [];
  for (var i = samePartsLength; i < fromParts.length; i++) {
    outputParts.push('..');
  }
  outputParts = outputParts.concat(toParts.slice(samePartsLength));
  return outputParts.join('/');
}};

var TTY = {ttys:[],init:function () {

```

```

// https://github.com/emscripten-core/emscripten/pull/1555
// if (ENVIRONMENT_IS_NODE) {
//   // currently, FS.init does not distinguish if process.stdin is a file or
TTY
//   // device, it always assumes it's a TTY device. because of this, we're
forcing
//   // process.stdin to UTF8 encoding to at least make stdin reading
compatible
//   // with text files until FS.init can be refactored.
//   process['stdin']['setEncoding']('utf8');
// }
},shutdown:function() {
// https://github.com/emscripten-core/emscripten/pull/1555
// if (ENVIRONMENT_IS_NODE) {
//   // inolen: any idea as to why node -e 'process.stdin.read()' wouldn't exit
immediately (with process.stdin being a tty)?
//   // isaacs: because now it's reading from the stream, you've expressed
interest in it, so that read() kicks off a _read() which creates a ReadReq operation
//   // inolen: I thought read() in that case was a synchronous operation that
just grabbed some amount of buffered data if it exists?
//   // isaacs: it is. but it also triggers a _read() call, which calls
readStart() on the handle
//   // isaacs: do process.stdin.pause() and i'd think it'd probably close the
pending call
//   process['stdin']['pause']();
// }
},register:function(dev, ops) {
TTY.ttys[dev] = { input: [], output: [], ops: ops };
FS.registerDevice(dev, TTY.stream_ops);
},stream_ops:{open:function(stream) {
var tty = TTY.ttys[stream.node.rdev];
if (!tty) {
throw new FS.ErrnoError(43);
}
stream.tty = tty;
stream.seekable = false;
},close:function(stream) {
// flush any pending line data
stream.tty.ops.flush(stream.tty);
},flush:function(stream) {
stream.tty.ops.flush(stream.tty);
},read:function(stream, buffer, offset, length, pos /* ignored */) {
if (!stream.tty || !stream.tty.ops.get_char) {
throw new FS.ErrnoError(60);
}
var bytesRead = 0;
for (var i = 0; i < length; i++) {
var result;
try {
result = stream.tty.ops.get_char(stream.tty);
} catch (e) {
throw new FS.ErrnoError(29);
}
if (result === undefined && bytesRead === 0) {
throw new FS.ErrnoError(6);
}
}
},write:function(stream, buffer, offset, length, pos) {
if (!stream.tty || !stream.tty.ops.set_char) {
throw new FS.ErrnoError(61);
}
var written = 0;
for (var i = 0; i < length; i++) {
var result;
try {
result = stream.tty.ops.set_char(stream.tty, buffer[offset + i]);
} catch (e) {
throw new FS.ErrnoError(29);
}
if (result === undefined) {
written++;
}
}
},seek:function(stream, offset, whence) {
if (!stream.tty || !stream.tty.ops.seek) {
throw new FS.ErrnoError(62);
}
var result;
try {
result = stream.tty.ops.seek(stream.tty, offset, whence);
} catch (e) {
throw new FS.ErrnoError(29);
}
if (result === undefined) {
return -1;
}
return result;
},truncate:function(stream, offset) {
if (!stream.tty || !stream.tty.ops.truncate) {
throw new FS.ErrnoError(63);
}
var result;
try {
result = stream.tty.ops.truncate(stream.tty, offset);
} catch (e) {
throw new FS.ErrnoError(29);
}
if (result === undefined) {
return -1;
}
return result;
},stat:function(stream) {
if (!stream.tty || !stream.tty.ops.stat) {
throw new FS.ErrnoError(64);
}
var result;
try {
result = stream.tty.ops.stat(stream.tty);
} catch (e) {
throw new FS.ErrnoError(29);
}
if (result === undefined) {
return -1;
}
return result;
},fsync:function(stream) {
if (!stream.tty || !stream.tty.ops.fsync) {
throw new FS.ErrnoError(65);
}
var result;
try {
result = stream.tty.ops.fsync(stream.tty);
} catch (e) {
throw new FS.ErrnoError(29);
}
if (result === undefined) {
return -1;
}
return result;
},fdatasync:function(stream) {
if (!stream.tty || !stream.tty.ops.fdatasync) {
throw new FS.ErrnoError(66);
}
var result;
try {
result = stream.tty.ops.fdatasync(stream.tty);
} catch (e) {
throw new FS.ErrnoError(29);
}
if (result === undefined) {
return -1;
}
return result;
},poll:function(stream, events, done) {
if (!stream.tty || !stream.tty.ops.poll) {
throw new FS.ErrnoError(67);
}
var result;
try {
result = stream.tty.ops.poll(stream.tty, events);
} catch (e) {
throw new FS.ErrnoError(29);
}
if (result === undefined) {
done(-1);
}
done(result);
}
}
}

```

```

        }
        if (result === null || result === undefined) break;
        bytesRead++;
        buffer[offset+i] = result;
    }
    if (bytesRead) {
        stream.node.timestamp = Date.now();
    }
    return bytesRead;
},write:function(stream, buffer, offset, length, pos) {
    if (!stream.tty || !stream.tty.ops.put_char) {
        throw new FS.ErrnoError(60);
    }
    try {
        for (var i = 0; i < length; i++) {
            stream.tty.ops.put_char(stream.tty, buffer[offset+i]);
        }
    } catch (e) {
        throw new FS.ErrnoError(29);
    }
    if (length) {
        stream.node.timestamp = Date.now();
    }
    return i;
}},default_tty_ops:{get_char:function(tty) {
    if (!tty.input.length) {
        var result = null;
        if (ENVIRONMENT_IS_NODE) {
            // we will read data by chunks of BUFSIZE
            var BUFSIZE = 256;
            var buf = Buffer.alloc(BUFSIZE);
            var bytesRead = 0;

            try {
                bytesRead = fs.readSync(process.stdin.fd, buf, 0, BUFSIZE, -1);
            } catch(e) {
                // Cross-platform differences: on Windows, reading EOF throws an
                // exception, but on other OSes,
                // reading EOF returns 0. Uniformize behavior by treating the EOF
                // exception to return 0.
                if (e.toString().includes('EOF')) bytesRead = 0;
                else throw e;
            }

            if (bytesRead > 0) {
                result = buf.slice(0, bytesRead).toString('utf-8');
            } else {
                result = null;
            }
        } else
            if (typeof window != 'undefined' &&
                typeof window.prompt == 'function') {
                // Browser.
                result = window.prompt('Input: ');
                // returns null on cancel
                if (result !== null) {

```

```

        result += '\n';
    }
} else if (typeof readline == 'function') {
    // Command line.
    result = readline();
    if (result !== null) {
        result += '\n';
    }
}
if (!result) {
    return null;
}
tty.input = intArrayFromString(result, true);
}
return tty.input.shift();
},put_char:function(tty, val) {
    if (val === null || val === 10) {
        out(UTF8ArrayToString(tty.output, 0));
        tty.output = [];
    } else {
        if (val != 0) tty.output.push(val); // val == 0 would cut text output off
in the middle.
    }
},flush:function(tty) {
    if (tty.output && tty.output.length > 0) {
        out(UTF8ArrayToString(tty.output, 0));
        tty.output = [];
    }
},default_tty1_ops:{put_char:function(tty, val) {
    if (val === null || val === 10) {
        err(UTF8ArrayToString(tty.output, 0));
        tty.output = [];
    } else {
        if (val != 0) tty.output.push(val);
    }
},flush:function(tty) {
    if (tty.output && tty.output.length > 0) {
        err(UTF8ArrayToString(tty.output, 0));
        tty.output = [];
    }
}
}}};

function zeroMemory(address, size) {
    HEAPU8.fill(0, address, address + size);
}

function alignMemory(size, alignment) {
    assert(alignment, "alignment argument is required");
    return Math.ceil(size / alignment) * alignment;
}
function mmapAlloc(size) {
    size = alignMemory(size, 65536);
    var ptr = _emscripten_builtin_memalign(65536, size);
    if (!ptr) return 0;
    zeroMemory(ptr, size);
}
```

```

        return ptr;
    }
var MEMFS = {ops_table:null,mount:function(mount) {
    return MEMFS.createNode(null, '/', 16384 | 511 /* 0777 */, 0);
},createNode:function(parent, name, mode, dev) {
    if (FS.isBlkdev(mode) || FS.isFIFO(mode)) {
        // no supported
        throw new FS.ErrnoError(63);
    }
    if (!MEMFS.ops_table) {
        MEMFS.ops_table = {
            dir: {
                node: {
                    getattr: MEMFS.node_opsgetattr,
                    setattr: MEMFS.node_opssetattr,
                    lookup: MEMFS.node_opslookup,
                    mknod: MEMFS.node_opsmknod,
                    rename: MEMFS.node_opsrename,
                    unlink: MEMFS.node_opsunlink,
                    rmdir: MEMFS.node_opsrmdir,
                    readdir: MEMFS.node_opsreaddir,
                    symlink: MEMFS.node_opssymlink
                },
                stream: {
                    llseek: MEMFS.stream_ops.llseek
                }
            },
            file: {
                node: {
                    getattr: MEMFS.node_opsgetattr,
                    setattr: MEMFS.node_opssetattr
                },
                stream: {
                    llseek: MEMFS.stream_ops.llseek,
                    read: MEMFS.stream_ops.read,
                    write: MEMFS.stream_ops.write,
                    allocate: MEMFS.stream_ops.allocate,
                    mmap: MEMFS.stream_ops.mmap,
                    msync: MEMFS.stream_ops.msync
                }
            },
            link: {
                node: {
                    getattr: MEMFS.node_opsgetattr,
                    setattr: MEMFS.node_opssetattr,
                    readlink: MEMFS.node_opsreadlink
                },
                stream: {}
            },
            chrdev: {
                node: {
                    getattr: MEMFS.node_opsgetattr,
                    setattr: MEMFS.node_opssetattr
                },
                stream: FS.chrdev_stream_ops
            }
        }
    }
}

```

```

        }
    };
}
var node = FS.createNode(parent, name, mode, dev);
if (FS.isDir(node.mode)) {
    node.node_ops = MEMFS.ops_table.dir.node;
    node.stream_ops = MEMFS.ops_table.dir.stream;
    node.contents = {};
} else if (FS.isFile(node.mode)) {
    node.node_ops = MEMFS.ops_table.file.node;
    node.stream_ops = MEMFS.ops_table.file.stream;
    node.usedBytes = 0; // The actual number of bytes used in the typed array, as
opposed to contents.length which gives the whole capacity.
    // When the byte data of the file is populated, this will point to either a
typed array, or a normal JS array. Typed arrays are preferred
    // for performance, and used by default. However, typed arrays are not
resizable like normal JS arrays are, so there is a small disk size
    // penalty involved for appending file writes that continuously grow a file
similar to std::vector capacity vs used -scheme.
    node.contents = null;
} else if (FS.isLink(node.mode)) {
    node.node_ops = MEMFS.ops_table.link.node;
    node.stream_ops = MEMFS.ops_table.link.stream;
} else if (FS.isChrdev(node.mode)) {
    node.node_ops = MEMFS.ops_table.chrdev.node;
    node.stream_ops = MEMFS.ops_table.chrdev.stream;
}
node.timestamp = Date.now();
// add the new node to the parent
if (parent) {
    parent.contents[name] = node;
    parent.timestamp = node.timestamp;
}
return node;
}, getFileDataAsTypedArray:function(node) {
    if (!node.contents) return new Uint8Array(0);
    if (node.contents.subarray) return node.contents.subarray(0, node.usedBytes);
// Make sure to not return excess unused bytes.
    return new Uint8Array(node.contents);
}, expandFileStorage:function(node, newCapacity) {
    var prevCapacity = node.contents ? node.contents.length : 0;
    if (prevCapacity >= newCapacity) return; // No need to expand, the storage was
already large enough.
    // Don't expand strictly to the given requested limit if it's only a very small
increase, but instead geometrically grow capacity.
    // For small filesizes (<1MB), perform size*2 geometric increase, but for large
sizes, do a much more conservative size*1.125 increase to
    // avoid overshooting the allocation cap by a very large margin.
    var CAPACITY_DOUBLING_MAX = 1024 * 1024;
    newCapacity = Math.max(newCapacity, (prevCapacity * (prevCapacity <
CAPACITY_DOUBLING_MAX ? 2.0 : 1.125)) >>> 0);
    if (prevCapacity != 0) newCapacity = Math.max(newCapacity, 256); // At minimum
allocate 256b for each file when expanding.
    var oldContents = node.contents;
    node.contents = new Uint8Array(newCapacity); // Allocate new storage.
}

```

```

    if (node.usedBytes > 0) node.contents.set(oldContents.subarray(0,
node.usedBytes), 0); // Copy old data over to the new storage.
},resizeFileStorage:function(node, newSize) {
    if (node.usedBytes == newSize) return;
    if (newSize == 0) {
        node.contents = null; // Fully decommit when requesting a resize to zero.
        node.usedBytes = 0;
    } else {
        var oldContents = node.contents;
        node.contents = new Uint8Array(newSize); // Allocate new storage.
        if (oldContents) {
            node.contents.set(oldContents.subarray(0, Math.min(newSize,
node.usedBytes))); // Copy old data over to the new storage.
        }
        node.usedBytes = newSize;
    }
},node_ops:{getattr:function(node) {
    var attr = {};
    // device numbers reuse inode numbers.
    attr.dev = FS.isChrdev(node.mode) ? node.id : 1;
    attr.ino = node.id;
    attr.mode = node.mode;
    attr.nlink = 1;
    attr.uid = 0;
    attr.gid = 0;
    attr.rdev = node.rdev;
    if (FS.isDir(node.mode)) {
        attr.size = 4096;
    } else if (FS.isFile(node.mode)) {
        attr.size = node.usedBytes;
    } else if (FS.isLink(node.mode)) {
        attr.size = node.link.length;
    } else {
        attr.size = 0;
    }
    attr.atime = new Date(node.timestamp);
    attr.mtime = new Date(node.timestamp);
    attr.ctime = new Date(node.timestamp);
    // NOTE: In our implementation, st_blocks = Math.ceil(st_size/st_blksize),
    //       but this is not required by the standard.
    attr.blksize = 4096;
    attr.blocks = Math.ceil(attr.size / attr.blksize);
    return attr;
},setattr:function(node, attr) {
    if (attr.mode !== undefined) {
        node.mode = attr.mode;
    }
    if (attr.timestamp !== undefined) {
        node.timestamp = attr.timestamp;
    }
    if (attr.size !== undefined) {
        MEMFS.resizeFileStorage(node, attr.size);
    }
},lookup:function(parent, name) {
    throw FS.genericErrors[44];
}
}

```

```

},mknod:function(parent, name, mode, dev) {
    return MEMFS.createNode(parent, name, mode, dev);
},rename:function(old_node, new_dir, new_name) {
    // if we're overwriting a directory at new_name, make sure it's empty.
    if (FS.isDir(old_node.mode)) {
        var new_node;
        try {
            new_node = FS.lookupNode(new_dir, new_name);
        } catch (e) {
        }
        if (new_node) {
            for (var i in new_node.contents) {
                throw new FS.ErrnoError(55);
            }
        }
    }
    // do the internal rewiring
    delete old_node.parent.contents[old_node.name];
    old_node.parent.timestamp = Date.now();
    old_node.name = new_name;
    new_dir.contents[new_name] = old_node;
    new_dir.timestamp = old_node.parent.timestamp;
    old_node.parent = new_dir;
},unlink:function(parent, name) {
    delete parent.contents[name];
    parent.timestamp = Date.now();
},rmdir:function(parent, name) {
    var node = FS.lookupNode(parent, name);
    for (var i in node.contents) {
        throw new FS.ErrnoError(55);
    }
    delete parent.contents[name];
    parent.timestamp = Date.now();
},readdir:function(node) {
    var entries = ['.','.'];
    for (var key in node.contents) {
        if (!node.contents.hasOwnProperty(key)) {
            continue;
        }
        entries.push(key);
    }
    return entries;
},symlink:function(parent, newname, oldpath) {
    var node = MEMFS.createNode(parent, newname, 511 /* 0777 */ | 40960, 0);
    node.link = oldpath;
    return node;
},readlink:function(node) {
    if (!FS.isLink(node.mode)) {
        throw new FS.ErrnoError(28);
    }
    return node.link;
},stream_ops:{read:function(stream, buffer, offset, length, position) {
    var contents = stream.node.contents;
    if (position >= stream.node.usedBytes) return 0;
    var size = Math.min(stream.node.usedBytes - position, length);

```

```

    assert(size >= 0);
    if (size > 8 && contents.subarray) { // non-trivial, and typed array
        buffer.set(contents.subarray(position, position + size), offset);
    } else {
        for (var i = 0; i < size; i++) buffer[offset + i] = contents[position + i];
    }
    return size;
}, write: function(stream, buffer, offset, length, position, canOwn) {
    // The data buffer should be a typed array view
    assert(!(buffer instanceof ArrayBuffer));
    // If the buffer is located in main memory (HEAP), and if
    // memory can grow, we can't hold on to references of the
    // memory buffer, as they may get invalidated. That means we
    // need to do copy its contents.
    if (buffer.buffer === HEAP8.buffer) {
        canOwn = false;
    }

    if (!length) return 0;
    var node = stream.node;
    node.timestamp = Date.now();

    if (buffer.subarray && (!node.contents || node.contents.subarray)) { // This
write is from a typed array to a typed array?
        if (canOwn) {
            assert(position === 0, 'canOwn must imply no weird position inside the
file');
            node.contents = buffer.subarray(offset, offset + length);
            node.usedBytes = length;
            return length;
        } else if (node.usedBytes === 0 && position === 0) { // If this is a simple
first write to an empty file, do a fast set since we don't need to care about old data.
            node.contents = buffer.slice(offset, offset + length);
            node.usedBytes = length;
            return length;
        } else if (position + length <= node.usedBytes) { // Writing to an already
allocated and used subrange of the file?
            node.contents.set(buffer.subarray(offset, offset + length), position);
            return length;
        }
    }

    // Appending to an existing file and we need to reallocate, or source data
did not come as a typed array.
    MEMFS.expandFileStorage(node, position+length);
    if (node.contents.subarray && buffer.subarray) {
        // Use typed array write which is available.
        node.contents.set(buffer.subarray(offset, offset + length), position);
    } else {
        for (var i = 0; i < length; i++) {
            node.contents[position + i] = buffer[offset + i]; // Or fall back to
manual write if not.
        }
    }
}

```

```

node.usedBytes = Math.max(node.usedBytes, position + length);
    return length;
}, llseek:function(stream, offset, whence) {
    var position = offset;
    if (whence === 1) {
        position += stream.position;
    } else if (whence === 2) {
        if (FS.isFile(stream.node.mode)) {
            position += stream.node.usedBytes;
        }
    }
    if (position < 0) {
        throw new FS.ErrnoError(28);
    }
    return position;
}, allocate:function(stream, offset, length) {
    MEMFS.expandFileStorage(stream.node, offset + length);
    stream.node.usedBytes = Math.max(stream.node.usedBytes, offset + length);
}, mmap:function(stream, address, length, position, prot, flags) {
    if (address !== 0) {
        // We don't currently support location hints for the address of the mapping
        throw new FS.ErrnoError(28);
    }
    if (!FS.isFile(stream.node.mode)) {
        throw new FS.ErrnoError(43);
    }
    var ptr;
    var allocated;
    var contents = stream.node.contents;
    // Only make a new copy when MAP_PRIVATE is specified.
    if (!(flags & 2) && contents.buffer === buffer) {
        // We can't emulate MAP_SHARED when the file is not backed by the buffer
        // we're mapping to (e.g. the HEAP buffer).
        allocated = false;
        ptr = contents.byteOffset;
    } else {
        // Try to avoid unnecessary slices.
        if (position > 0 || position + length < contents.length) {
            if (contents.subarray) {
                contents = contents.subarray(position, position + length);
            } else {
                contents = Array.prototype.slice.call(contents, position, position +
length);
            }
        }
        allocated = true;
        ptr = mmapAlloc(length);
        if (!ptr) {
            throw new FS.ErrnoError(48);
        }
        HEAP8.set(contents, ptr);
    }
    return { ptr: ptr, allocated: allocated };
}, msync:function(stream, buffer, offset, length, mmapFlags) {
    if (!FS.isFile(stream.node.mode)) {

```

```

        throw new FS.ErrnoError(43);
    }
    if (mmapFlags & 2) {
        // MAP_PRIVATE calls need not to be synced back to underlying fs
        return 0;
    }

    var bytesWritten = MEMFS.stream_ops.write(stream, buffer, 0, length, offset,
false);
    // should we check if bytesWritten and length are the same?
    return 0;
}};

/** @param {boolean=} noRunDep */
function asyncLoad(url, onload, onerror, noRunDep) {
    var dep = !noRunDep ? getUniqueRunDependency('al ' + url) : '';
    readAsync(url, function(arrayBuffer) {
        assert(arrayBuffer, 'Loading data file "' + url + '" failed (no
arrayBuffer).');
        onload(new Uint8Array(arrayBuffer));
        if (dep) removeRunDependency(dep);
    }, function(event) {
        if (onerror) {
            onerror();
        } else {
            throw 'Loading data file "' + url + '" failed.';
        }
    });
    if (dep) addRunDependency(dep);
}

var IDBFS = {dbs:{},indexedDB:() => {
    if (typeof indexedDB != 'undefined') return indexedDB;
    var ret = null;
    if (typeof window == 'object') ret = window.indexedDB || window.mozIndexedDB ||
window.webkitIndexedDB || window.msIndexedDB;
    assert(ret, 'IDBFS used, but indexedDB not supported');
    return ret;
},DB_VERSION:21,DB_STORE_NAME:"FILE_DATA",mount:function(mount) {
    // reuse all of the core MEMFS functionality
    return MEMFS.mount.apply(null, arguments);
},syncfs:(mount, populate, callback) => {
    IDBFS.getLocalSet(mount, (err, local) => {
        if (err) return callback(err);

        IDBFS.getRemoteSet(mount, (err, remote) => {
            if (err) return callback(err);

            var src = populate ? remote : local;
            var dst = populate ? local : remote;

            IDBFS.reconcile(src, dst, callback);
        });
    });
},getDB:(name, callback) => {
}
}

```

```

// check the cache first
var db = IDBFS.dbs[name];
if (db) {
  return callback(null, db);
}

var req;
try {
  req = IDBFS.indexedDB().open(name, IDBFS.DB_VERSION);
} catch (e) {
  return callback(e);
}
if (!req) {
  return callback("Unable to connect to IndexedDB");
}
req.onupgradeneeded = (e) => {
  var db = /** @type {IDBDatabase} */ (e.target.result);
  var transaction = e.target.transaction;

  var fileStore;

  if (db.objectStoreNames.contains(IDBFS.DB_STORE_NAME)) {
    fileStore = transaction.objectStore(IDBFS.DB_STORE_NAME);
  } else {
    fileStore = db.createObjectStore(IDBFS.DB_STORE_NAME);
  }

  if (!fileStore.indexNames.contains('timestamp')) {
    fileStore.createIndex('timestamp', 'timestamp', { unique: false });
  }
};
req.onsuccess = () => {
  db = /** @type {IDBDatabase} */ (req.result);

  // add to the cache
  IDBFS.dbs[name] = db;
  callback(null, db);
};

req.onerror = (e) => {
  callback(this.error);
  e.preventDefault();
};

},getLocalSet:(mount, callback) => {
  var entries = {};

  function isRealDir(p) {
    return p !== '.' && p !== '..';
  };
  function toAbsolute(root) {
    return (p) => {
      return PATH.join2(root, p);
    }
  };
}

```

```

    var check =
FS.readdir(mount.mountpoint).filter(isRealDir).map(toAbsolute(mount.mountpoint));

    while (check.length) {
        var path = check.pop();
        var stat;

        try {
            stat = FS.stat(path);
        } catch (e) {
            return callback(e);
        }

        if (FS.isDir(stat.mode)) {
            check.push.apply(check,
FS.readdir(path).filter(isRealDir).map(toAbsolute(path)));
        }

        entries[path] = { 'timestamp': stat.mtime };
    }

    return callback(null, { type: 'local', entries: entries });
},getRemoteSet:(mount, callback) => {
    var entries = {};

    IDBFS.getDB(mount.mountpoint, (err, db) => {
        if (err) return callback(err);

        try {
            var transaction = db.transaction([IDBFS.DB_STORE_NAME], 'readonly');
            transaction.onerror = (e) => {
                callback(this.error);
                e.preventDefault();
            };

            var store = transaction.objectStore(IDBFS.DB_STORE_NAME);
            var index = store.index('timestamp');

            index.openKeyCursor().onsuccess = (event) => {
                var cursor = event.target.result;

                if (!cursor) {
                    return callback(null, { type: 'remote', db: db, entries: entries });
                }

                entries[cursor.primaryKey] = { 'timestamp': cursor.key };

                cursor.continue();
            };
        } catch (e) {
            return callback(e);
        }
    });
},loadLocalEntry:(path, callback) => {
    var stat, node;

```

```

try {
  var lookup = FS.lookupPath(path);
  node = lookup.node;
  stat = FS.stat(path);
} catch (e) {
  return callback(e);
}

if (FS.isDir(stat.mode)) {
  return callback(null, { 'timestamp': stat.mtime, 'mode': stat.mode });
} else if (FS.isFile(stat.mode)) {
  // Performance consideration: storing a normal JavaScript array to a
  // IndexedDB is much slower than storing a typed array.
  // Therefore always convert the file contents to a typed array first before
  // writing the data to IndexedDB.
  node.contents = MEMFS.getFileDataAsTypedArray(node);
  return callback(null, { 'timestamp': stat.mtime, 'mode': stat.mode,
  'contents': node.contents });
} else {
  return callback(new Error('node type not supported'));
}
},storeLocalEntry:(path, entry, callback) => {
try {
  if (FS.isDir(entry['mode'])) {
    FS.mkdirTree(path, entry['mode']);
  } else if (FS.isFile(entry['mode'])) {
    FS.writeFile(path, entry['contents'], { canOwn: true });
  } else {
    return callback(new Error('node type not supported'));
  }
  FS.chmod(path, entry['mode']);
  FS.utime(path, entry['timestamp'], entry['timestamp']);
} catch (e) {
  return callback(e);
}
callback(null);
},removeLocalEntry:(path, callback) => {
try {
  var lookup = FS.lookupPath(path);
  var stat = FS.stat(path);

  if (FS.isDir(stat.mode)) {
    FS.rmdir(path);
  } else if (FS.isFile(stat.mode)) {
    FS.unlink(path);
  }
} catch (e) {
  return callback(e);
}
callback(null);
},loadRemoteEntry:(store, path, callback) => {
var req = store.get(path);

```

```

req.onsuccess = (event) => { callback(null, event.target.result); };
req.onerror = (e) => {
  callback(this.error);
  e.preventDefault();
};
}, storeRemoteEntry:(store, path, entry, callback) => {
  try {
    var req = store.put(entry, path);
  } catch (e) {
    callback(e);
    return;
  }
  req.onsuccess = () => { callback(null); };
  req.onerror = (e) => {
    callback(this.error);
    e.preventDefault();
  };
}, removeRemoteEntry:(store, path, callback) => {
  var req = store.delete(path);
  req.onsuccess = () => { callback(null); };
  req.onerror = (e) => {
    callback(this.error);
    e.preventDefault();
  };
}, reconcile:(src, dst, callback) => {
  var total = 0;

  var create = [];
  Object.keys(src.entries).forEach(function (key) {
    var e = src.entries[key];
    var e2 = dst.entries[key];
    if (!e2 || e['timestamp'].getTime() != e2['timestamp'].getTime()) {
      create.push(key);
      total++;
    }
  });

  var remove = [];
  Object.keys(dst.entries).forEach(function (key) {
    if (!src.entries[key]) {
      remove.push(key);
      total++;
    }
  });

  if (!total) {
    return callback(null);
  }

  var errored = false;
  var db = src.type === 'remote' ? src.db : dst.db;
  var transaction = db.transaction([IDBFS.DB_STORE_NAME], 'readwrite');
  var store = transaction.objectStore(IDBFS.DB_STORE_NAME);

  function done(err) {

```

```

    if (err && !errored) {
      errored = true;
      return callback(err);
    }
  };

transaction.onerror = (e) => {
  done(this.error);
  e.preventDefault();
};

transaction.oncomplete = (e) => {
  if (!errored) {
    callback(null);
  }
};

// sort paths in ascending order so directory entries are created
// before the files inside them
create.sort().forEach((path) => {
  if (dst.type === 'local') {
    IDBFS.loadRemoteEntry(store, path, (err, entry) => {
      if (err) return done(err);
      IDBFS.storeLocalEntry(path, entry, done);
    });
  } else {
    IDBFS.loadLocalEntry(path, (err, entry) => {
      if (err) return done(err);
      IDBFS.storeRemoteEntry(store, path, entry, done);
    });
  }
});

// sort paths in descending order so files are deleted before their
// parent directories
remove.sort().reverse().forEach((path) => {
  if (dst.type === 'local') {
    IDBFS.removeLocalEntry(path, done);
  } else {
    IDBFS.removeRemoteEntry(store, path, done);
  }
});
};

```

```

var ERRNO_MESSAGES = {0:"Success",1:"Arg list too long",2:"Permission denied",3:"Address already in use",4:"Address not available",5:"Address family not supported by protocol family",6:"No more processes",7:"Socket already connected",8:"Bad file number",9:"Trying to read unreadable message",10:"Mount device busy",11:"Operation canceled",12:"No children",13:"Connection aborted",14:"Connection refused",15:"Connection reset by peer",16:"File locking deadlock error",17:"Destination address required",18:"Math arg out of domain of func",19:"Quota exceeded",20:"File exists",21:"Bad address",22:"File too large",23:"Host is unreachable",24:"Identifier removed",25:"Illegal byte sequence",26:"Connection already in progress",27:"Interrupted system call",28:"Invalid argument",29:"I/O error",30:"Socket is already connected",31:"Is a directory",32:"Too many symbolic links",33:"Too many open files",34:"Too many links",35:"Message too long",36:"Multihop attempted",37:"File or path name too long",38:"Network interface is not configured",39:"Connection reset by network",40:"Network is unreachable",41:"Too many open files in system",42:"No buffer space available",43:"No such device",44:"No such file or directory",45:"Exec format error",46:"No record locks available",47:"The link has been severed",48:"Not enough core",49:"No message of desired type",50:"Protocol not available",51:"No space left on device",52:"Function not implemented",53:"Socket is not connected",54:"Not a directory",55:"Directory not empty",56:"State not recoverable",57:"Socket operation on non-socket",59:"Not a typewriter",60:"No such device or address",61:"Value too large for defined data type",62:"Previous owner died",63:"Not super-user",64:"Broken pipe",65:"Protocol error",66:"Unknown protocol",67:"Protocol wrong type for socket",68:"Math result not representable",69:"Read only file system",70:"Illegal seek",71:"No such process",72:"Stale file handle",73:"Connection timed out",74:"Text file busy",75:"Cross-device link",100:"Device not a stream",101:"Bad font file fmt",102:"Invalid slot",103:"Invalid request code",104:"No anode",105:"Block device required",106:"Channel number out of range",107:"Level 3 halted",108:"Level 3 reset",109:"Link number out of range",110:"Protocol driver not attached",111:"No CSI structure available",112:"Level 2 halted",113:"Invalid exchange",114:"Invalid request descriptor",115:"Exchange full",116:"No data (for no delay io)",117:"Timer expired",118:"Out of streams resources",119:"Machine is not on the network",120:"Package not installed",121:"The object is remote",122:"Advertise error",123:"Srmount error",124:"Communication error on send",125:"Cross mount point (not really error)",126:"Given log. name not unique",127:"f.d. invalid for this operation",128:"Remote address changed",129:"Can't access a needed shared lib",130:"Accessing a corrupted shared lib",131:".lib section in a.out corrupted",132:"Attempting to link in too many libs",133:"Attempting to exec a shared library",135:"Streams pipe error",136:"Too many users",137:"Socket type not supported",138:"Not supported",139:"Protocol family not supported",140:"Can't send after socket shutdown",141:"Too many references",142:"Host is down",148:"No medium (in tape drive)",156:"Level 2 not synchronized"};

```

```

var ERRNO_CODES = {};
var FS =
{root:null,mounts:[],devices:{},streams:[],nextInode:1,nameTable:null,currentPath:"/",initialized:false,ignorePermissions:true,ErrnoError:null,genericErrors:{},filesystems:null,syncFSRequests:0,lookupPath:(path, opts = {}) => {
    path = PATH_FS.resolve(FS.cwd(), path);

    if (!path) return { path: '', node: null };

    var defaults = {
        follow_mount: true,
        recurse_count: 0
    };
    opts = Object.assign(defaults, opts)

    if (opts.recurse_count > 8) { // max recursive lookup of 8
        throw new FS.ErrnoError(32);
    }
}

```

```
}

// split the path
var parts = PATH.normalizeArray(path.split('/').filter((p) => !!p), false);

// start at the root
var current = FS.root;
var current_path = '/';

for (var i = 0; i < parts.length; i++) {
```

```

var islast = (i === parts.length-1);
if (islast && opts.parent) {
    // stop resolving
    break;
}

current = FS.lookupNode(current, parts[i]);
current_path = PATH.join2(current_path, parts[i]);

// jump to the mount's root node if this is a mountpoint
if (FS.isMountpoint(current)) {
    if (!islast || (islast && opts.follow_mount)) {
        current = current.mounted.root;
    }
}

// by default, lookupPath will not follow a symlink if it is the final path
// component.
// setting opts.follow = true will override this behavior.
if (!islast || opts.follow) {
    var count = 0;
    while (FS.isLink(current.mode)) {
        var link = FS.readlink(current_path);
        current_path = PATH_FS.resolve(PATH.dirname(current_path), link);

        var lookup = FS.lookupPath(current_path, { recurse_count:
opts.recurse_count + 1 });
        current = lookup.node;

        if (count++ > 40) { // limit max consecutive symlinks to 40
(SYMLOOP_MAX).
            throw new FS.ErrnoError(32);
        }
    }
}

return { path: current_path, node: current };
}, getPath:(node) => {
    var path;
    while (true) {
        if (FS.isRoot(node)) {
            var mount = node.mount.mountpoint;
            if (!path) return mount;
            return mount[mount.length-1] !== '/' ? mount + '/' + path : mount + path;
        }
        path = path ? node.name + '/' + path : node.name;
        node = node.parent;
    }
}, hashName:(parentid, name) => {
    var hash = 0;

    for (var i = 0; i < name.length; i++) {
        hash = ((hash << 5) - hash + name.charCodeAt(i)) | 0;
    }
}

```

```

    return ((parentid + hash) >>> 0) % FS.nameTable.length;
},hashAddNode:(node) => {
  var hash = FS.hashName(node.parent.id, node.name);
  node.name_next = FS.nameTable[hash];
  FS.nameTable[hash] = node;
},hashRemoveNode:(node) => {
  var hash = FS.hashName(node.parent.id, node.name);
  if (FS.nameTable[hash] === node) {
    FS.nameTable[hash] = node.name_next;
  } else {
    var current = FS.nameTable[hash];
    while (current) {
      if (current.name_next === node) {
        current.name_next = node.name_next;
        break;
      }
      current = current.name_next;
    }
  }
},lookupNode:(parent, name) => {
  var errCode = FS.mayLookup(parent);
  if (errCode) {
    throw new FS.ErrnoError(errCode, parent);
  }
  var hash = FS.hashName(parent.id, name);
  for (var node = FS.nameTable[hash]; node; node = node.name_next) {
    var nodeName = node.name;
    if (node.parent.id === parent.id && nodeName === name) {
      return node;
    }
  }
  // if we failed to find it in the cache, call into the VFS
  return FS.lookup(parent, name);
},createNode:(parent, name, mode, rdev) => {
  assert(typeof parent === 'object')
  var node = new FS.FSNode(parent, name, mode, rdev);

  FS.hashAddNode(node);

  return node;
},destroyNode:(node) => {
  FS.hashRemoveNode(node);
},isRoot:(node) => {
  return node === node.parent;
},isMountpoint:(node) => {
  return !node.mounted;
},isFile:(mode) => {
  return (mode & 61440) === 32768;
},isDir:(mode) => {
  return (mode & 61440) === 16384;
},isLink:(mode) => {
  return (mode & 61440) === 40960;
},isChrdev:(mode) => {
  return (mode & 61440) === 8192;
},isBlkdev:(mode) => {

```

```

        return (mode & 61440) === 24576;
    }, isFIFO: (mode) => {
        return (mode & 61440) === 4096;
    }, isSocket: (mode) => {
        return (mode & 49152) === 49152;
}

}, flagModes:{ "r":0, "r+":2, "w":577, "w+":578, "a":1089, "a+":1090 }, modeStringToFlags:(str)
=> {
    var flags = FS.flagModes[str];
    if (typeof flags == 'undefined') {
        throw new Error('Unknown file open mode: ' + str);
    }
    return flags;
}, flagsToPermissionString:(flag) => {
    var perms = ['r', 'w', 'rw'][flag & 3];
    if ((flag & 512)) {
        perms += 'w';
    }
    return perms;
}, nodePermissions:(node, perms) => {
    if (FS.ignorePermissions) {
        return 0;
    }
    // return 0 if any user, group or owner bits are set.
    if (perms.includes('r') && !(node.mode & 292)) {
        return 2;
    } else if (perms.includes('w') && !(node.mode & 146)) {
        return 2;
    } else if (perms.includes('x') && !(node.mode & 73)) {
        return 2;
    }
    return 0;
}, mayLookup:(dir) => {
    var errCode = FS.nodePermissions(dir, 'x');
    if (errCode) return errCode;
    if (!dir.node_ops.lookup) return 2;
    return 0;
}, mayCreate:(dir, name) => {
    try {
        var node = FS.lookupNode(dir, name);
        return 20;
    } catch (e) {
    }
    return FS.nodePermissions(dir, 'wx');
}, mayDelete:(dir, name, isdir) => {
    var node;
    try {
        node = FS.lookupNode(dir, name);
    } catch (e) {
        return e(errno);
    }
    var errCode = FS.nodePermissions(dir, 'wx');
    if (errCode) {
        return errCode;
    }
    if (isdir) {

```

```

        if (!FS.isDir(node.mode)) {
            return 54;
        }
        if (FS.isRoot(node) || FS.getPath(node) === FS.cwd()) {
            return 10;
        }
    } else {
        if (FS.isDir(node.mode)) {
            return 31;
        }
    }
    return 0;
},mayOpen:(node, flags) => {
    if (!node) {
        return 44;
    }
    if (FS.isLink(node.mode)) {
        return 32;
    } else if (FS.isDir(node.mode)) {
        if (FS.flagsToString(flags) !== 'r' || // opening for write
            (flags & 512)) { // TODO: check for O_SEARCH? (== search for dir only)
            return 31;
        }
    }
    return FS.nodePermissions(node, FS.flagsToString(flags));
},MAX_OPEN_FDS:4096,nextfd:(fd_start = 0, fd_end = FS.MAX_OPEN_FDS) => {
    for (var fd = fd_start; fd <= fd_end; fd++) {
        if (!FS.streams[fd]) {
            return fd;
        }
    }
    throw new FS.ErrnoError(33);
},getStream:(fd) => FS.streams[fd],createStream:(stream, fd_start, fd_end) => {
    if (!FS.FSStream) {
        FS.FSStream = /** @constructor */ function() {};
        FS.FSStream.prototype = {
            object: {
                get: function() { return this.node; },
                set: function(val) { this.node = val; }
            },
            isRead: {
                get: function() { return (this.flags & 2097155) !== 1; }
            },
            isWrite: {
                get: function() { return (this.flags & 2097155) !== 0; }
            },
            isAppend: {
                get: function() { return (this.flags & 1024); }
            }
        };
    }
    // clone it, so we can return an instance of FSStream
    stream = Object.assign(new FS.FSStream(), stream);
    var fd = FS.nextfd(fd_start, fd_end);
    stream.fd = fd;
}

```

```

    FS.streams[fd] = stream;
    return stream;
}, closeStream:(fd) => {
    FS.streams[fd] = null;
}, chrdev_stream_ops:{open:(stream) => {
    var device = FS.getDevice(stream.node.rdev);
    // override node's stream ops with the device's
    stream.stream_ops = device.stream_ops;
    // forward the open call
    if (stream.stream_ops.open) {
        stream.stream_ops.open(stream);
    }
}, lseek:() => {
    throw new FS.ErrnoError(70);
}, major:(dev) => ((dev) >> 8), minor:(dev) => ((dev) & 0xff), makedev:(ma, mi)
=> ((ma) << 8 | (mi)), registerDevice:(dev, ops) => {
    FS.devices[dev] = { stream_ops: ops };
}, getDevice:(dev) => FS.devices[dev], getMounts:(mount) => {
    var mounts = [];
    var check = [mount];

    while (check.length) {
        var m = check.pop();

        mounts.push(m);

        check.push.apply(check, m.mounts);
    }

    return mounts;
}, syncfs:(populate, callback) => {
    if (typeof populate == 'function') {
        callback = populate;
        populate = false;
    }
}

FS.syncFSRequests++;

if (FS.syncFSRequests > 1) {
    err('warning: ' + FS.syncFSRequests + ' FS.syncfs operations in flight at
once, probably just doing extra work');
}

var mounts = FS.getMounts(FS.root.mount);
var completed = 0;

function doCallback(errCode) {
    assert(FS.syncFSRequests > 0);
    FS.syncFSRequests--;
    return callback(errCode);
}

function done(errCode) {
    if (errCode) {
        if (!done.errorred) {

```

```

        done.errorred = true;
        return doCallback(errCode);
    }
    return;
}
if (++completed >= mounts.length) {
    doCallback(null);
}
};

// sync all mounts
mounts.forEach((mount) => {
    if (!mount.type.syncfs) {
        return done(null);
    }
    mount.type.syncfs(mount, populate, done);
});
},mount:(type, opts, mountpoint) => {
    if (typeof type == 'string') {
        // The filesystem was not included, and instead we have an error
        // message stored in the variable.
        throw type;
    }
    var root = mountpoint === '/';
    var pseudo = !mountpoint;
    var node;

    if (root && FS.root) {
        throw new FS.ErrnoError(10);
    } else if (!root && !pseudo) {
        var lookup = FS.lookupPath(mountpoint, { follow_mount: false });

        mountpoint = lookup.path; // use the absolute path
        node = lookup.node;

        if (FS.isMountpoint(node)) {
            throw new FS.ErrnoError(10);
        }
    }

    if (!FS.isdir(node.mode)) {
        throw new FS.ErrnoError(54);
    }
}

var mount = {
    type: type,
    opts: opts,
    mountpoint: mountpoint,
    mounts: []
};

// create a root node for the fs
var mountRoot = type.mount(mount);
mountRoot.mount = mount;
mount.root = mountRoot;

```

```

if (root) {
  FS.root = mountRoot;
} else if (node) {
  // set as a mountpoint
  node.mounted = mount;

  // add the new mount to the current mount's children
  if (node.mount) {
    node.mount.mounts.push(mount);
  }
}

return mountRoot;
},unmount:(mountpoint) => {
  var lookup = FS.lookupPath(mountpoint, { follow_mount: false });

  if (!FS.isMountpoint(lookup.node)) {
    throw new FS.ErrnoError(28);
  }

  // destroy the nodes for this mount, and all its child mounts
  var node = lookup.node;
  var mount = node.mounted;
  var mounts = FS.getMounts(mount);

  Object.keys(FS.nameTable).forEach((hash) => {
    var current = FS.nameTable[hash];

    while (current) {
      var next = current.name_next;

      if (mounts.includes(current.mount)) {
        FS.destroyNode(current);
      }

      current = next;
    }
  });

  // no longer a mountpoint
  node.mounted = null;

  // remove this mount from the child mounts
  var idx = node.mount.mounts.indexOf(mount);
  assert(idx !== -1);
  node.mount.mounts.splice(idx, 1);
},lookup:(parent, name) => {
  return parent.node_ops.lookup(parent, name);
},mknod:(path, mode, dev) => {
  var lookup = FS.lookupPath(path, { parent: true });
  var parent = lookup.node;
  var name = PATH.basename(path);
  if (!name || name === '.' || name === '..') {
    throw new FS.ErrnoError(28);
}

```

```

    }
    var errCode = FS.mayCreate(parent, name);
    if (errCode) {
        throw new FS.ErrnoError(errCode);
    }
    if (!parent.node_ops.mknod) {
        throw new FS.ErrnoError(63);
    }
    return parent.node_ops.mknod(parent, name, mode, dev);
},create:(path, mode) => {
    mode = mode !== undefined ? mode : 438 /* 0666 */;
    mode &= 4095;
    mode |= 32768;
    return FS.mknod(path, mode, 0);
},mkdir:(path, mode) => {
    mode = mode !== undefined ? mode : 511 /* 0777 */;
    mode &= 511 | 512;
    mode |= 16384;
    return FS.mknod(path, mode, 0);
},mkdirTree:(path, mode) => {
    var dirs = path.split('/');
    var d = '';
    for (var i = 0; i < dirs.length; ++i) {
        if (!dirs[i]) continue;
        d += '/' + dirs[i];
        try {
            FS.mkdir(d, mode);
        } catch(e) {
            if (e.errno != 20) throw e;
        }
    }
},mkdev:(path, mode, dev) => {
    if (typeof dev == 'undefined') {
        dev = mode;
        mode = 438 /* 0666 */;
    }
    mode |= 8192;
    return FS.mknod(path, mode, dev);
},symlink:(oldpath, newpath) => {
    if (!PATH_FS.resolve(oldpath)) {
        throw new FS.ErrnoError(44);
    }
    var lookup = FS.lookupPath(newpath, { parent: true });
    var parent = lookup.node;
    if (!parent) {
        throw new FS.ErrnoError(44);
    }
    var newname = PATH.basename(newpath);
    var errCode = FS.mayCreate(parent, newname);
    if (errCode) {
        throw new FS.ErrnoError(errCode);
    }
    if (!parent.node_ops.symlink) {
        throw new FS.ErrnoError(63);
    }
}

```

```

    return parent.node_ops.symlink(parent, newname, oldpath);
}, rename:(old_path, new_path) => {
  var old_dirname = PATH.dirname(old_path);
  var new_dirname = PATH.dirname(new_path);
  var old_name = PATH.basename(old_path);
  var new_name = PATH.basename(new_path);
  // parents must exist
  var lookup, old_dir, new_dir;

  // let the errors from non existant directories percolate up
  lookup = FS.lookupPath(old_path, { parent: true });
  old_dir = lookup.node;
  lookup = FS.lookupPath(new_path, { parent: true });
  new_dir = lookup.node;

  if (!old_dir || !new_dir) throw new FS.ErrnoError(44);
  // need to be part of the same mount
  if (old_dir.mount !== new_dir.mount) {
    throw new FS.ErrnoError(75);
  }
  // source must exist
  var old_node = FS.lookupNode(old_dir, old_name);
  // old path should not be an ancestor of the new path
  var relative = PATH_FS.relative(old_path, new_dirname);
  if (relative.charAt(0) !== '.') {
    throw new FS.ErrnoError(28);
  }
  // new path should not be an ancestor of the old path
  relative = PATH_FS.relative(new_path, old_dirname);
  if (relative.charAt(0) !== '.') {
    throw new FS.ErrnoError(55);
  }
  // see if the new path already exists
  var new_node;
  try {
    new_node = FS.lookupNode(new_dir, new_name);
  } catch (e) {
    // not fatal
  }
  // early out if nothing needs to change
  if (old_node === new_node) {
    return;
  }
  // we'll need to delete the old entry
  var isdir = FS.isDir(old_node.mode);
  var errCode = FS.mayDelete(old_dir, old_name, isdir);
  if (errCode) {
    throw new FS.ErrnoError(errCode);
  }
  // need delete permissions if we'll be overwriting.
  // need create permissions if new doesn't already exist.
  errCode = new_node ?
    FS.mayDelete(new_dir, new_name, isdir) :
    FS.mayCreate(new_dir, new_name);
  if (errCode) {

```

```

        throw new FS.ErrnoError(errCode);
    }
    if (!old_dir.node_ops.rename) {
        throw new FS.ErrnoError(63);
    }
    if (FS.isMountpoint(old_node) || (new_node && FS.isMountpoint(new_node))) {
        throw new FS.ErrnoError(10);
    }
    // if we are going to change the parent, check write permissions
    if (new_dir !== old_dir) {
        errCode = FS.nodePermissions(old_dir, 'w');
        if (errCode) {
            throw new FS.ErrnoError(errCode);
        }
    }
    // remove the node from the lookup hash
    FS.hashRemoveNode(old_node);
    // do the underlying fs rename
    try {
        old_dir.node_ops.rename(old_node, new_dir, new_name);
    } catch (e) {
        throw e;
    } finally {
        // add the node back to the hash (in case node_ops.rename
        // changed its name)
        FS.hashAddNode(old_node);
    }
}, rmdir: (path) => {
    var lookup = FS.lookupPath(path, { parent: true });
    var parent = lookup.node;
    var name = PATH.basename(path);
    var node = FS.lookupNode(parent, name);
    var errCode = FS.mayDelete(parent, name, true);
    if (errCode) {
        throw new FS.ErrnoError(errCode);
    }
    if (!parent.node_ops.rmdir) {
        throw new FS.ErrnoError(63);
    }
    if (FS.isMountpoint(node)) {
        throw new FS.ErrnoError(10);
    }
    parent.node_ops.rmdir(parent, name);
    FS.destroyNode(node);
}, readdir: (path) => {
    var lookup = FS.lookupPath(path, { follow: true });
    var node = lookup.node;
    if (!node.node_ops.readdir) {
        throw new FS.ErrnoError(54);
    }
    return node.node_ops.readdir(node);
}, unlink: (path) => {
    var lookup = FS.lookupPath(path, { parent: true });
    var parent = lookup.node;
    if (!parent) {

```

```

        throw new FS.ErrnoError(44);
    }
    var name = PATH.basename(path);
    var node = FS.lookupNode(parent, name);
    var errCode = FS.mayDelete(parent, name, false);
    if (errCode) {
        // According to POSIX, we should map EISDIR to EPERM, but
        // we instead do what Linux does (and we must, as we use
        // the musl linux libc).
        throw new FS.ErrnoError(errCode);
    }
    if (!parent.node_ops.unlink) {
        throw new FS.ErrnoError(63);
    }
    if (FS.isMountpoint(node)) {
        throw new FS.ErrnoError(10);
    }
    parent.node_ops.unlink(parent, name);
    FS.destroyNode(node);
},readlink:(path) => {
    var lookup = FS.lookupPath(path);
    var link = lookup.node;
    if (!link) {
        throw new FS.ErrnoError(44);
    }
    if (!link.node_ops.readlink) {
        throw new FS.ErrnoError(28);
    }
    return PATH_FS.resolve(FS.getPath(link.parent), link.node_ops.readlink(link));
},stat:(path, dontFollow) => {
    var lookup = FS.lookupPath(path, { follow: !dontFollow });
    var node = lookup.node;
    if (!node) {
        throw new FS.ErrnoError(44);
    }
    if (!node.node_ops.getattr) {
        throw new FS.ErrnoError(63);
    }
    return node.node_ops.getattr(node);
},lstat:(path) => {
    return FS.stat(path, true);
},chmod:(path, mode, dontFollow) => {
    var node;
    if (typeof path == 'string') {
        var lookup = FS.lookupPath(path, { follow: !dontFollow });
        node = lookup.node;
    } else {
        node = path;
    }
    if (!node.node_ops.setattr) {
        throw new FS.ErrnoError(63);
    }
    node.node_ops.setattr(node, {
        mode: (mode & 4095) | (node.mode & ~4095),
        timestamp: Date.now()
    })
}

```

```

    });
}, lchmod: (path, mode) => {
  FS.chmod(path, mode, true);
}, fchmod: (fd, mode) => {
  var stream = FS.getStream(fd);
  if (!stream) {
    throw new FS.ErrnoError(8);
  }
  FS.chmod(stream.node, mode);
}, chown: (path, uid, gid, dontFollow) => {
  var node;
  if (typeof path == 'string') {
    var lookup = FS.lookupPath(path, { follow: !dontFollow });
    node = lookup.node;
  } else {
    node = path;
  }
  if (!node.node_ops.setattr) {
    throw new FS.ErrnoError(63);
  }
  node.node_opssetattr(node, {
    timestamp: Date.now()
    // we ignore the uid / gid for now
  });
}, lchown: (path, uid, gid) => {
  FS.chown(path, uid, gid, true);
}, fchown: (fd, uid, gid) => {
  var stream = FS.getStream(fd);
  if (!stream) {
    throw new FS.ErrnoError(8);
  }
  FS.chown(stream.node, uid, gid);
}, truncate: (path, len) => {
  if (len < 0) {
    throw new FS.ErrnoError(28);
  }
  var node;
  if (typeof path == 'string') {
    var lookup = FS.lookupPath(path, { follow: true });
    node = lookup.node;
  } else {
    node = path;
  }
  if (!node.node_ops.setattr) {
    throw new FS.ErrnoError(63);
  }
  if (FS.isDir(node.mode)) {
    throw new FS.ErrnoError(31);
  }
  if (!FS.isFile(node.mode)) {
    throw new FS.ErrnoError(28);
  }
  var errCode = FS.nodePermissions(node, 'w');
  if (errCode) {
    throw new FS.ErrnoError(errCode);
  }
}

```

```

    }
    node.node_ops.setattr(node, {
      size: len,
      timestamp: Date.now()
    });
  }, ftruncate:(fd, len) => {
    var stream = FS.getStream(fd);
    if (!stream) {
      throw new FS.ErrnoError(8);
    }
    if ((stream.flags & 2097155) === 0) {
      throw new FS.ErrnoError(28);
    }
    FS.truncate(stream.node, len);
  }, utime:(path, atime, mtime) => {
    var lookup = FS.lookupPath(path, { follow: true });
    var node = lookup.node;
    node.node_ops.setattr(node, {
      timestamp: Math.max(atime, mtime)
    });
  }, open:(path, flags, mode, fd_start, fd_end) => {
    if (path === "") {
      throw new FS.ErrnoError(44);
    }
    flags = typeof flags == 'string' ? FS.modeStringToFlags(flags) : flags;
    mode = typeof mode == 'undefined' ? 438 /* 0666 */ : mode;
    if ((flags & 64)) {
      mode = (mode & 4095) | 32768;
    } else {
      mode = 0;
    }
    var node;
    if (typeof path == 'object') {
      node = path;
    } else {
      path = PATH.normalize(path);
      try {
        var lookup = FS.lookupPath(path, {
          follow: !(flags & 131072)
        });
        node = lookup.node;
      } catch (e) {
        // ignore
      }
    }
    // perhaps we need to create the node
    var created = false;
    if ((flags & 64)) {
      if (node) {
        // if O_CREAT and O_EXCL are set, error out if the node already exists
        if ((flags & 128)) {
          throw new FS.ErrnoError(20);
        }
      } else {
        // node doesn't exist, try to create it
      }
    }
  }
}

```

```

        node = FS.mknod(path, mode, 0);
        created = true;
    }
}
if (!node) {
    throw new FS.ErrnoError(44);
}
// can't truncate a device
if (FS.isChrdev(node.mode)) {
    flags &= ~512;
}
// if asked only for a directory, then this must be one
if ((flags & 65536) && !FS.isDir(node.mode)) {
    throw new FS.ErrnoError(54);
}
// check permissions, if this is not a file we just created now (it is ok to
// create and write to a file with read-only permissions; it is read-only
// for later use)
if (!created) {
    var errCode = FS.mayOpen(node, flags);
    if (errCode) {
        throw new FS.ErrnoError(errCode);
    }
}
// do truncation if necessary
if ((flags & 512)) {
    FS.truncate(node, 0);
}
// we've already handled these, don't pass down to the underlying vfs
flags &= ~(128 | 512 | 131072);

// register the stream with the filesystem
var stream = FS.createStream({
    node: node,
    path: FS.getPath(node), // we want the absolute path to the node
    flags: flags,
    seekable: true,
    position: 0,
    stream_ops: node.stream_ops,
    // used by the file family libc calls (fopen, fwrite, perror, etc.)
    ungotten: [],
    error: false
}, fd_start, fd_end);
// call the new stream's open function
if (stream.stream_ops.open) {
    stream.stream_ops.open(stream);
}
if (Module['logReadFiles'] && !(flags & 1)) {
    if (!FS.readFiles) FS.readFiles = {};
    if (!(path in FS.readFiles)) {
        FS.readFiles[path] = 1;
    }
}
return stream;
}, close:(stream) => {

```

```

if (FS.isClosed(stream)) {
    throw new FS.ErrnoError(8);
}
if (stream.getdents) stream.getdents = null; // free readdir state
try {
    if (stream.stream_ops.close) {
        stream.stream_ops.close(stream);
    }
} catch (e) {
    throw e;
} finally {
    FS.closeStream(stream.fd);
}
stream.fd = null;
},isClosed:(stream) => {
    return stream.fd === null;
},llseek:(stream, offset, whence) => {
    if (FS.isClosed(stream)) {
        throw new FS.ErrnoError(8);
    }
    if (!stream.seekable || !stream.stream_ops.llseek) {
        throw new FS.ErrnoError(70);
    }
    if (whence != 0 && whence != 1 && whence != 2) {
        throw new FS.ErrnoError(28);
    }
    stream.position = stream.stream_ops.llseek(stream, offset, whence);
    stream.ungotten = [];
    return stream.position;
},read:(stream, buffer, offset, length, position) => {
    if (length < 0 || position < 0) {
        throw new FS.ErrnoError(28);
    }
    if (FS.isClosed(stream)) {
        throw new FS.ErrnoError(8);
    }
    if ((stream.flags & 2097155) === 1) {
        throw new FS.ErrnoError(8);
    }
    if (FS.isDir(stream.node.mode)) {
        throw new FS.ErrnoError(31);
    }
    if (!stream.stream_ops.read) {
        throw new FS.ErrnoError(28);
    }
    var seeking = typeof position != 'undefined';
    if (!seeking) {
        position = stream.position;
    } else if (!stream.seekable) {
        throw new FS.ErrnoError(70);
    }
    var bytesRead = stream.stream_ops.read(stream, buffer, offset, length,
position);
    if (!seeking) stream.position += bytesRead;
    return bytesRead;
}

```

```

},write:(stream, buffer, offset, length, position, canOwn) => {
  if (length < 0 || position < 0) {
    throw new FS.ErrnoError(28);
  }
  if (FS.isClosed(stream)) {
    throw new FS.ErrnoError(8);
  }
  if ((stream.flags & 2097155) === 0) {
    throw new FS.ErrnoError(8);
  }
  if (FS.isDir(stream.node.mode)) {
    throw new FS.ErrnoError(31);
  }
  if (!stream.stream_ops.write) {
    throw new FS.ErrnoError(28);
  }
  if (stream.seekable && stream.flags & 1024) {
    // seek to the end before writing in append mode
    FS.llseek(stream, 0, 2);
  }
  var seeking = typeof position != 'undefined';
  if (!seeking) {
    position = stream.position;
  } else if (!stream.seekable) {
    throw new FS.ErrnoError(70);
  }
  var bytesWritten = stream.stream_ops.write(stream, buffer, offset, length,
position, canOwn);
  if (!seeking) stream.position += bytesWritten;
  return bytesWritten;
},allocate:(stream, offset, length) => {
  if (FS.isClosed(stream)) {
    throw new FS.ErrnoError(8);
  }
  if (offset < 0 || length <= 0) {
    throw new FS.ErrnoError(28);
  }
  if ((stream.flags & 2097155) === 0) {
    throw new FS.ErrnoError(8);
  }
  if (!FS.isFile(stream.node.mode) && !FS.isDir(stream.node.mode)) {
    throw new FS.ErrnoError(43);
  }
  if (!stream.stream_ops.allocate) {
    throw new FS.ErrnoError(138);
  }
  stream.stream_ops.allocate(stream, offset, length);
},mmap:(stream, address, length, position, prot, flags) => {
  // User requests writing to file (prot & PROT_WRITE != 0).
  // Checking if we have permissions to write to the file unless
  // MAP_PRIVATE flag is set. According to POSIX spec it is possible
  // to write to file opened in read-only mode with MAP_PRIVATE flag,
  // as all modifications will be visible only in the memory of
  // the current process.
  if ((prot & 2) !== 0

```

```

    && (flags & 2) === 0
    && (stream.flags & 2097155) !== 2) {
        throw new FS.ErrnoError(2);
    }
    if ((stream.flags & 2097155) === 1) {
        throw new FS.ErrnoError(2);
    }
    if (!stream.stream_ops.mmap) {
        throw new FS.ErrnoError(43);
    }
    return stream.stream_ops.mmap(stream, address, length, position, prot, flags);
},msync:(stream, buffer, offset, length, mmapFlags) => {
    if (!stream || !stream.stream_ops.msync) {
        return 0;
    }
    return stream.stream_ops.msync(stream, buffer, offset, length, mmapFlags);
},munmap:(stream) => 0,ioctl:(stream, cmd, arg) => {
    if (!stream.stream_ops.ioctl) {
        throw new FS.ErrnoError(59);
    }
    return stream.stream_ops.ioctl(stream, cmd, arg);
},readFile:(path, opts = {}) => {
    opts.flags = opts.flags || 0;
    opts.encoding = opts.encoding || 'binary';
    if (opts.encoding !== 'utf8' && opts.encoding !== 'binary') {
        throw new Error('Invalid encoding type "' + opts.encoding + '"');
    }
    var ret;
    var stream = FS.open(path, opts.flags);
    var stat = FS.stat(path);
    var length = stat.size;
    var buf = new Uint8Array(length);
    FS.read(stream, buf, 0, length, 0);
    if (opts.encoding === 'utf8') {
        ret = UTF8ArrayToString(buf, 0);
    } else if (opts.encoding === 'binary') {
        ret = buf;
    }
    FS.close(stream);
    return ret;
},writeFile:(path, data, opts = {}) => {
    opts.flags = opts.flags || 577;
    var stream = FS.open(path, opts.flags, opts.mode);
    if (typeof data === 'string') {
        var buf = new Uint8Array(lengthBytesUTF8(data)+1);
        var actualNumBytes = stringToUTF8Array(data, buf, 0, buf.length);
        FS.write(stream, buf, 0, actualNumBytes, undefined, opts.canOwn);
    } else if (ArrayBuffer.isView(data)) {
        FS.write(stream, data, 0, data.byteLength, undefined, opts.canOwn);
    } else {
        throw new Error('Unsupported data type');
    }
    FS.close(stream);
},cwd:() => FS.currentPath,chdir:(path) => {
    var lookup = FS.lookupPath(path, { follow: true });

```

```

if (lookup.node === null) {
  throw new FS.ErrnoError(44);
}
if (!FS.isDir(lookup.node.mode)) {
  throw new FS.ErrnoError(54);
}
var errCode = FS.nodePermissions(lookup.node, 'x');
if (errCode) {
  throw new FS.ErrnoError(errCode);
}
FS.currentPath = lookup.path;
},createDefaultDirectories:() => {
  FS.mkdir('/tmp');
  FS.mkdir('/home');
  FS.mkdir('/home/web_user');
},createDefaultDevices:() => {
  // create /dev
  FS.mkdir('/dev');
  // setup /dev/null
  FS.registerDevice(FS.makedev(1, 3), {
    read: () => 0,
    write: (stream, buffer, offset, length, pos) => length,
  });
  FS.mkdev('/dev/null', FS.makedev(1, 3));
  // setup /dev/tty and /dev/tty1
  // stderr needs to print output using err() rather than out()
  // so we register a second tty just for it.
  TTY.register(FS.makedev(5, 0), TTY.default_tty_ops);
  TTY.register(FS.makedev(6, 0), TTY.default_tty1_ops);
  FS.mkdev('/dev/tty', FS.makedev(5, 0));
  FS.mkdev('/dev/tty1', FS.makedev(6, 0));
  // setup /dev/[u]random
  var random_device = getRandomDevice();
  FS.createDevice('/dev', 'random', random_device);
  FS.createDevice('/dev', 'urandom', random_device);
  // we're not going to emulate the actual shm device,
  // just create the tmp dirs that reside in it commonly
  FS.mkdir('/dev/shm');
  FS.mkdir('/dev/shm/tmp');
},createSpecialDirectories:() => {
  // create /proc/self/fd which allows /proc/self/fd/6 => readlink gives the
  // name of the stream for fd 6 (see test_unistd_ttynname)
  FS.mkdir('/proc');
  var proc_self = FS.mkdir('/proc/self');
  FS.mkdir('/proc/self/fd');
  FS.mount({
    mount: () => {
      var node = FS.createNode(proc_self, 'fd', 16384 | 511 /* 0777 */, 73);
      node.node_ops = {
        lookup: (parent, name) => {
          var fd = +name;
          var stream = FS.getStream(fd);
          if (!stream) throw new FS.ErrnoError(8);
          var ret = {
            parent: null,

```

```

        mount: { mountpoint: 'fake' },
        node_ops: { readlink: () => stream.path },
    };
    ret.parent = ret; // make it look like a simple root node
    return ret;
}
};

return node;
},
{}, '/proc/self/fd');
}, createStandardStreams:() => {
// TODO deprecate the old functionality of a single
// input / output callback and that utilizes FS.createDevice
// and instead require a unique set of stream ops

// by default, we symlink the standard streams to the
// default tty devices. however, if the standard streams
// have been overwritten we create a unique device for
// them instead.
if (Module['stdin']) {
    FS.createDevice('/dev', 'stdin', Module['stdin']);
} else {
    FS.symlink('/dev/tty', '/dev/stdin');
}
if (Module['stdout']) {
    FS.createDevice('/dev', 'stdout', null, Module['stdout']);
} else {
    FS.symlink('/dev/tty', '/dev/stdout');
}
if (Module['stderr']) {
    FS.createDevice('/dev', 'stderr', null, Module['stderr']);
} else {
    FS.symlink('/dev/tty1', '/dev/stderr');
}

// open default streams for the stdin, stdout and stderr devices
var stdin = FS.open('/dev/stdin', 0);
var stdout = FS.open('/dev/stdout', 1);
var stderr = FS.open('/dev/stderr', 1);
assert(stdin.fd === 0, 'invalid handle for stdin (' + stdin.fd + ')');
assert(stdout.fd === 1, 'invalid handle for stdout (' + stdout.fd + ')');
assert(stderr.fd === 2, 'invalid handle for stderr (' + stderr.fd + ')');
}, ensureErrnoError:() => {
    if (FS.ErrnoError) return;
    FS.ErrnoError = /** @this{Object} */ function ErrnoError(errno, node) {
        this.node = node;
        this.setErrno = /** @this{Object} */ function(errno) {
            this errno = errno;
            for (var key in ERRNO_CODES) {
                if (ERRNO_CODES[key] === errno) {
                    this.code = key;
                    break;
                }
            }
        };
    };
};
```

```

        this.setErrno(errno);
        this.message = ERRNO_MESSAGES[errno];

        // Try to get a maximally helpful stack trace. On Node.js, getting
Error.stack
            // now ensures it shows what we want.
            if (this.stack) {
                // Define the stack property for Node.js 4, which otherwise errors on the
next line.
                Object.defineProperty(this, "stack", { value: (new Error).stack, writable:
true });
                this.stack = demangleAll(this.stack);
            }
        };
        FS.ErrnoError.prototype = new Error();
        FS.ErrnoError.prototype.constructor = FS.ErrnoError;
        // Some errors may happen quite a bit, to avoid overhead we reuse them (and
suffer a lack of stack info)
        [44].forEach((code) => {
            FS.genericErrors[code] = new FS.ErrnoError(code);
            FS.genericErrors[code].stack = '<generic error, no stack>';
        });
    },staticInit:() => {
    FS.ensureErrnoError();

    FS.nameTable = new Array(4096);

    FS.mount(MEMFS, {}, '/');
    FS.createDefaultDirectories();
    FS.createDefaultDevices();
    FS.createSpecialDirectories();

    FS.filesystems = {
        'MEMFS': MEMFS,
        'IDBFS': IDBFS,
    };
},init:(input, output, error) => {
    assert(!FS.init.initialized, 'FS.init was previously called. If you want to
initialize later with custom parameters, remove any earlier calls (note that one is
automatically added to the generated code)');
    FS.init.initialized = true;

    FS.ensureErrnoError();

    // Allow Module.stdin etc. to provide defaults, if none explicitly passed to us
here
    Module['stdin'] = input || Module['stdin'];
    Module['stdout'] = output || Module['stdout'];
    Module['stderr'] = error || Module['stderr'];

    FS.createStandardStreams();
},quit:() => {
    FS.init.initialized = false;
    // Call musl-internal function to close all stdio streams, so nothing is

```

```

// left in internal buffers.
__stdio_exit();
// close all of our streams
for (var i = 0; i < FS.streams.length; i++) {
    var stream = FS.streams[i];
    if (!stream) {
        continue;
    }
    FS.close(stream);
}
},getMode:(canRead, canWrite) => {
    var mode = 0;
    if (canRead) mode |= 292 | 73;
    if (canWrite) mode |= 146;
    return mode;
},findObject:(path, dontResolveLastLink) => {
    var ret = FS.analyzePath(path, dontResolveLastLink);
    if (ret.exists) {
        return ret.object;
    } else {
        return null;
    }
},analyzePath:(path, dontResolveLastLink) => {
    // operate from within the context of the symlink's target
    try {
        var lookup = FS.lookupPath(path, { follow: !dontResolveLastLink });
        path = lookup.path;
    } catch (e) {
    }
    var ret = {
        isRoot: false, exists: false, error: 0, name: null, path: null, object: null,
        parentExists: false, parentPath: null, parentObject: null
    };
    try {
        var lookup = FS.lookupPath(path, { parent: true });
        ret.parentExists = true;
        ret.parentPath = lookup.path;
        ret.parentObject = lookup.node;
        ret.name = PATH.basename(path);
        lookup = FS.lookupPath(path, { follow: !dontResolveLastLink });
        ret.exists = true;
        ret.path = lookup.path;
        ret.object = lookup.node;
        ret.name = lookup.node.name;
        ret.isRoot = lookup.path === '/';
    } catch (e) {
        ret.error = e(errno);
    };
    return ret;
},createPath:(parent, path, canRead, canWrite) => {
    parent = typeof parent == 'string' ? parent : FS.getPath(parent);
    var parts = path.split('/').reverse();
    while (parts.length) {
        var part = parts.pop();
        if (!part) continue;

```

```

    var current = PATH.join2(parent, part);
    try {
        FS.mkdir(current);
    } catch (e) {
        // ignore EEXIST
    }
    parent = current;
}
return current;
},createFile:(parent, name, properties, canRead, canWrite) => {
    var path = PATH.join2(typeof parent == 'string' ? parent : FS.getPath(parent),
name);
    var mode = FS.getMode(canRead, canWrite);
    return FS.create(path, mode);
},createDataFile:(parent, name, data, canRead, canWrite, canOwn) => {
    var path = name;
    if (parent) {
        parent = typeof parent == 'string' ? parent : FS.getPath(parent);
        path = name ? PATH.join2(parent, name) : parent;
    }
    var mode = FS.getMode(canRead, canWrite);
    var node = FS.create(path, mode);
    if (data) {
        if (typeof data == 'string') {
            var arr = new Array(data.length);
            for (var i = 0, len = data.length; i < len; ++i) arr[i] =
data.charCodeAt(i);
            data = arr;
        }
        // make sure we can write to the file
        FS.chmod(node, mode | 146);
        var stream = FS.open(node, 577);
        FS.write(stream, data, 0, data.length, 0, canOwn);
        FS.close(stream);
        FS.chmod(node, mode);
    }
    return node;
},createDevice:(parent, name, input, output) => {
    var path = PATH.join2(typeof parent == 'string' ? parent : FS.getPath(parent),
name);
    var mode = FS.getMode(!input, !output);
    if (!FS.createDevice.major) FS.createDevice.major = 64;
    var dev = FS.makedev(FS.createDevice.major++, 0);
    // Create a fake device that a set of stream ops to emulate
    // the old behavior.
    FS.registerDevice(dev, {
        open: (stream) => {
            stream.seekable = false;
        },
        close: (stream) => {
            // flush any pending line data
            if (output && output.buffer && output.buffer.length) {
                output(10);
            }
        },
    });
}

```

```

read: (stream, buffer, offset, length, pos /* ignored */) => {
    var bytesRead = 0;
    for (var i = 0; i < length; i++) {
        var result;
        try {
            result = input();
        } catch (e) {
            throw new FS.ErrnoError(29);
        }
        if (result === undefined && bytesRead === 0) {
            throw new FS.ErrnoError(6);
        }
        if (result === null || result === undefined) break;
        bytesRead++;
        buffer[offset+i] = result;
    }
    if (bytesRead) {
        stream.node.timestamp = Date.now();
    }
    return bytesRead;
},
write: (stream, buffer, offset, length, pos) => {
    for (var i = 0; i < length; i++) {
        try {
            output(buffer[offset+i]);
        } catch (e) {
            throw new FS.ErrnoError(29);
        }
    }
    if (length) {
        stream.node.timestamp = Date.now();
    }
    return i;
});
return FS.mkdev(path, mode, dev);
}, forceLoadFile:(obj) => {
    if (obj.isDevice || obj.isFolder || obj.link || obj.contents) return true;
    if (typeof XMLHttpRequest != 'undefined') {
        throw new Error("Lazy loading should have been performed (contents set) in createLazyFile, but it was not. Lazy loading only works in web workers. Use --embed-file or --preload-file in emcc on the main thread.");
    } else if (read_) {
        // Command-line.
        try {
            // WARNING: Can't read binary files in V8's d8 or tracemonkey's js, as
            //          read() will try to parse UTF8.
            obj.contents = intArrayFromString(read_(obj.url), true);
            obj.usedBytes = obj.contents.length;
        } catch (e) {
            throw new FS.ErrnoError(29);
        }
    } else {
        throw new Error('Cannot load without read() or XMLHttpRequest.');
    }
}

```

```

},createLazyFile:(parent, name, url, canRead, canWrite) => {
    // Lazy chunked Uint8Array (implements get and length from Uint8Array). Actual
    getting is abstracted away for eventual reuse.
    /** @constructor */
    function LazyUint8Array() {
        this.lengthKnown = false;
        this.chunks = []; // Loaded chunks. Index is the chunk number
    }
    LazyUint8Array.prototype.get = /** @this{Object} */ function
LazyUint8Array_get(idx) {
    if (idx > this.length-1 || idx < 0) {
        return undefined;
    }
    var chunkOffset = idx % this.chunkSize;
    var chunkNum = (idx / this.chunkSize)|0;
    return this.getter(chunkNum)[chunkOffset];
};
    LazyUint8Array.prototype.setDataGetter = function
LazyUint8Array_setDataGetter(getter) {
    this.getter = getter;
};
    LazyUint8Array.prototype.cacheLength = function LazyUint8Array_cacheLength() {
        // Find length
        var xhr = new XMLHttpRequest();
        xhr.open('HEAD', url, false);
        xhr.send(null);
        if (!(xhr.status >= 200 && xhr.status < 300 || xhr.status === 304)) throw new
Error("Couldn't load " + url + ". Status: " + xhr.status);
        var datalength = Number(xhr.getResponseHeader("Content-length"));
        var header;
        var hasByteServing = (header = xhr.getResponseHeader("Accept-Ranges")) &&
header === "bytes";
        var usesGzip = (header = xhr.getResponseHeader("Content-Encoding")) && header
=== "gzip";

        var chunkSize = 1024*1024; // Chunk size in bytes

        if (!hasByteServing) chunkSize = datalength;

        // Function to get a range from the remote URL.
        var doXHR = (from, to) => {
            if (from > to) throw new Error("invalid range (" + from + ", " + to + ") or
no bytes requested!");
            if (to > datalength-1) throw new Error("only " + datalength + " bytes
available! programmer error!");

            // TODO: Use mozResponseArrayBuffer, responseStream, etc. if available.
            var xhr = new XMLHttpRequest();
            xhr.open('GET', url, false);
            if (datalength !== chunkSize) xhr.setRequestHeader("Range", "bytes=" + from
+ "-" + to);

            // Some hints to the browser that we want binary data.
            xhr.responseType = 'arraybuffer';
            if (xhr.overrideMimeType) {

```

```

        xhr.overrideMimeType('text/plain; charset=x-user-defined');
    }

    xhr.send(null);
    if (!(xhr.status >= 200 && xhr.status < 300 || xhr.status === 304)) throw
new Error("Couldn't load " + url + ". Status: " + xhr.status);
    if (xhr.response !== undefined) {
        return new Uint8Array/** @type{Array<number>} */(xhr.response || []);
    } else {
        return intArrayFromString(xhr.responseText || '', true);
    }
};

var lazyArray = this;
lazyArray.setDataGetter((chunkNum) => {
    var start = chunkNum * chunkSize;
    var end = (chunkNum+1) * chunkSize - 1; // including this byte
    end = Math.min(end, datalength-1); // if datalength-1 is selected, this is
the last block
    if (typeof lazyArray.chunks[chunkNum] == 'undefined') {
        lazyArray.chunks[chunkNum] = doXHR(start, end);
    }
    if (typeof lazyArray.chunks[chunkNum] == 'undefined') throw new
Error('doXHR failed!');
    return lazyArray.chunks[chunkNum];
});

if (usesGzip || !datalength) {
    // if the server uses gzip or doesn't supply the length, we have to
download the whole file to get the (uncompressed) length
    chunkSize = datalength = 1; // this will force getter(0)/doXHR do download
the whole file
    datalength = this.getter(0).length;
    chunkSize = datalength;
    out("LazyFiles on gzip forces download of the whole file when length is
accessed");
}

this._length = datalength;
this._chunkSize = chunkSize;
this.lengthKnown = true;
};

if (typeof XMLHttpRequest != 'undefined') {
    if (!ENVIRONMENT_IS_WORKER) throw 'Cannot do synchronous binary XHRs outside
webworkers in modern browsers. Use --embed-file or --preload-file in emcc';
    var lazyArray = new LazyUint8Array();
    Object.defineProperties(lazyArray, {
        length: {
            get: /** @this{Object} */ function() {
                if (!this.lengthKnown) {
                    this.cacheLength();
                }
                return this._length;
            }
        },
        chunkSize: {

```

```

        get: /** @this{Object} */ function() {
            if (!this.lengthKnown) {
                this.cacheLength();
            }
            return this._chunkSize;
        }
    });
}

var properties = { isDevice: false, contents: lazyArray };
} else {
    var properties = { isDevice: false, url: url };
}

var node = FS.createFile(parent, name, properties, canRead, canWrite);
// This is a total hack, but I want to get this lazy file code out of the
// core of MEMFS. If we want to keep this lazy file concept I feel it should
// be its own thin LAZYFS proxying calls to MEMFS.
if (properties.contents) {
    node.contents = properties.contents;
} else if (properties.url) {
    node.contents = null;
    node.url = properties.url;
}
// Add a function that defers querying the file size until it is asked the
first time.
Object.defineProperties(node, {
    usedBytes: {
        get: /** @this {FSNode} */ function() { return this.contents.length; }
    }
});
// override each stream op with one that tries to force load the lazy file
first
var stream_ops = {};
var keys = Object.keys(node.stream_ops);
keys.forEach((key) => {
    var fn = node.stream_ops[key];
    stream_ops[key] = function forceLoadLazyFile() {
        FS.forceLoadFile(node);
        return fn.apply(null, arguments);
    };
});
// use a custom read function
stream_ops.read = (stream, buffer, offset, length, position) => {
    FS.forceLoadFile(node);
    var contents = stream.node.contents;
    if (position >= contents.length)
        return 0;
    var size = Math.min(contents.length - position, length);
    assert(size >= 0);
    if (contents.slice) { // normal array
        for (var i = 0; i < size; i++) {
            buffer[offset + i] = contents[position + i];
        }
    } else {

```

```

        for (var i = 0; i < size; i++) { // LazyUint8Array from sync binary XHR
            buffer[offset + i] = contents.get(position + i);
        }
    }
    return size;
};

node.stream_ops = stream_ops;
return node;
},createPreloadedFile:(parent, name, url, canRead, canWrite, onload, onerror,
dontCreateFile, canOwn, preFinish) => {
    // TODO we should allow people to just pass in a complete filename instead
    // of parent and name being that we just join them anyways
    var fullname = name ? PATH_FS.resolve(PATH.join2(parent, name)) : parent;
    var dep = getUniqueRunDependency('cp ' + fullname); // might have several
active requests for the same fullname
    function processData(byteArray) {
        function finish(byteArray) {
            if (preFinish) preFinish();
            if (!dontCreateFile) {
                FS.createDataFile(parent, name, byteArray, canRead, canWrite, canOwn);
            }
            if (onload) onload();
            removeRunDependency(dep);
        }
        if (Browser.handledByPreloadPlugin(byteArray, fullname, finish, () => {
            if (onerror) onerror();
            removeRunDependency(dep);
        })) {
            return;
        }
        finish(byteArray);
    }
    addRunDependency(dep);
    if (typeof url == 'string') {
        asyncLoad(url, (byteArray) => processData(byteArray), onerror);
    } else {
        processData(url);
    }
},indexedDB:() => {
    return window.indexedDB || window.mozIndexedDB || window.webkitIndexedDB ||
window.msIndexedDB;
},DB_NAME:() => {
    return 'EM_FS_' + window.location.pathname;
},DB_VERSION:20,DB_STORE_NAME:"FILE_DATA",saveFilesToDB:(paths, onload, onerror)
=> {
    onload = onload || (() => {});
    onerror = onerror || (() => {});
    var indexedDB = FS.indexedDB();
    try {
        var openRequest = indexedDB.open(FS.DB_NAME(), FS.DB_VERSION);
    } catch (e) {
        return onerror(e);
    }
    openRequest.onupgradeneeded = () => {
        out('creating db');
    }
}

```

```

    var db = openRequest.result;
    db.createObjectStore(FS.DB_STORE_NAME);
};

openRequest.onload = () => {
    var db = openRequest.result;
    var transaction = db.transaction([FS.DB_STORE_NAME], 'readwrite');
    var files = transaction.objectStore(FS.DB_STORE_NAME);
    var ok = 0, fail = 0, total = paths.length;
    function finish() {
        if (fail == 0) onload(); else onerror();
    }
    paths.forEach((path) => {
        var putRequest = files.put(FS.analyzePath(path).object.contents, path);
        putRequest.onload = () => { ok++; if (ok + fail == total) finish() };
        putRequest.onerror = () => { fail++; if (ok + fail == total) finish() };
    });
    transaction.onerror = onerror;
};
openRequest.onerror = onerror;
}, loadFilesFromDB:(paths, onload, onerror) => {
    onload = onload || (() => {});
    onerror = onerror || (() => {});
    var indexedDB = FS.indexedDB();
    try {
        var openRequest = indexedDB.open(FS.DB_NAME(), FS.DB_VERSION);
    } catch (e) {
        return onerror(e);
    }
    openRequest.onupgradeneeded = onerror; // no database to load from
    openRequest.onload = () => {
        var db = openRequest.result;
        try {
            var transaction = db.transaction([FS.DB_STORE_NAME], 'readonly');
        } catch(e) {
            onerror(e);
            return;
        }
        var files = transaction.objectStore(FS.DB_STORE_NAME);
        var ok = 0, fail = 0, total = paths.length;
        function finish() {
            if (fail == 0) onload(); else onerror();
        }
        paths.forEach((path) => {
            var getRequest = files.get(path);
            getRequest.onload = () => {
                if (FS.analyzePath(path).exists) {
                    FS.unlink(path);
                }
                FS.createDataFile(PATH.dirname(path), PATH.basename(path),
getRequest.result, true, true, true);
                ok++;
                if (ok + fail == total) finish();
            };
            getRequest.onerror = () => { fail++; if (ok + fail == total) finish() };
        });
    });
}

```

```

        transaction.onerror = onerror;
    };
    openRequest.onerror = onerror;
}, absolutePath: () => {
    abort('FS.absolutePath has been removed; use PATH_FS.resolve instead');
}, createFolder: () => {
    abort('FS.createFolder has been removed; use FS.mkdir instead');
}, createLink: () => {
    abort('FS.createLink has been removed; use FS.symlink instead');
}, joinPath: () => {
    abort('FS.joinPath has been removed; use PATH.join instead');
}, mmapAlloc: () => {
    abort('FS.mmapAlloc has been replaced by the top level function mmapAlloc');
}, standardizePath: () => {
    abort('FS.standardizePath has been removed; use PATH.normalize instead');
};

var SYSCALLS = {DEFAULT_POLLMASK:5, calculateAt:function(dirfd, path, allowEmpty) {
    if (path[0] === '/') {
        return path;
    }
    // relative path
    var dir;
    if (dirfd === -100) {
        dir = FS.cwd();
    } else {
        var dirstream = FS.getStream(dirfd);
        if (!dirstream) throw new FS.ErrnoError(8);
        dir = dirstream.path;
    }
    if (path.length == 0) {
        if (!allowEmpty) {
            throw new FS.ErrnoError(44);
        }
        return dir;
    }
    return PATH.join2(dir, path);
}, doStat:function(func, path, buf) {
    try {
        var stat = func(path);
    } catch (e) {
        if (e && e.node && PATH.normalize(path) !==
PATH.normalize(FS.getPath(e.node))) {
            // an error occurred while trying to look up the path; we should just
report ENOTDIR
            return -54;
        }
        throw e;
    }
    HEAP32[((buf)>>2)] = stat.dev;
    HEAP32[((buf)+(4))>>2] = 0;
    HEAP32[((buf)+(8))>>2] = stat.ino;
    HEAP32[((buf)+(12))>>2] = stat.mode;
    HEAP32[((buf)+(16))>>2] = stat.nlink;
    HEAP32[((buf)+(20))>>2] = stat.uid;
    HEAP32[((buf)+(24))>>2] = stat.gid;
}

```

```

    HEAP32[((buf)+(28))>>2] = stat.rdev;
    HEAP32[((buf)+(32))>>2] = 0;
    (tempI64 = [stat.size>>>0,(tempDouble=stat.size,(+(Math.abs(tempDouble))) >=
1.0 ? (tempDouble > 0.0 ? ((Math.min((+(Math.floor((tempDouble)/4294967296.0))),4294967295.0))|0)>>>0 : (~~~((+(Math.ceil((tempDouble -
+((~~(tempDouble))))>>>0))/4294967296.0))))>>>0) : 0)],HEAP32[((buf)+(40))>>2] =
tempI64[0],HEAP32[((buf)+(44))>>2] = tempI64[1]);
    HEAP32[((buf)+(48))>>2] = 4096;
    HEAP32[((buf)+(52))>>2] = stat.blocks;
    HEAP32[((buf)+(56))>>2] = (stat.atime.getTime() / 1000)|0;
    HEAP32[((buf)+(60))>>2] = 0;
    HEAP32[((buf)+(64))>>2] = (stat.mtime.getTime() / 1000)|0;
    HEAP32[((buf)+(68))>>2] = 0;
    HEAP32[((buf)+(72))>>2] = (stat.ctime.getTime() / 1000)|0;
    HEAP32[((buf)+(76))>>2] = 0;
    (tempI64 = [stat.ino>>>0,(tempDouble=stat.ino,(+(Math.abs(tempDouble))) >= 1.0
? (tempDouble > 0.0 ? ((Math.min((+(Math.floor((tempDouble)/4294967296.0))),4294967295.0))|0)>>>0 : (~~~((+(Math.ceil((tempDouble -
+((~~(tempDouble))))>>>0))/4294967296.0))))>>>0) : 0)],HEAP32[((buf)+(80))>>2] =
tempI64[0],HEAP32[((buf)+(84))>>2] = tempI64[1]);
    return 0;
},doMsync:function(addr, stream, len, flags, offset) {
    var buffer = HEAPU8.slice(addr, addr + len);
    FS.msync(stream, buffer, offset, len, flags);
},doMkdir:function(path, mode) {
    // remove a trailing slash, if one - /a/b/ has basename of '', but
    // we want to create b in the context of this function
    path = PATH.normalize(path);
    if (path[path.length-1] === '/') path = path.substr(0, path.length-1);
    FS.mkdir(path, mode, 0);
    return 0;
},doMknod:function(path, mode, dev) {
    // we don't want this in the JS API as it uses mknod to create all nodes.
    switch (mode & 61440) {
        case 32768:
        case 8192:
        case 24576:
        case 4096:
        case 49152:
            break;
        default: return -28;
    }
    FS.mknod(path, mode, dev);
    return 0;
},doReadlink:function(path, buf, bufsize) {
    if (bufsize <= 0) return -28;
    var ret = FS.readlink(path);

    var len = Math.min(bufsize, lengthBytesUTF8(ret));
    var endChar = HEAP8[buf+len];
    stringToUTF8(ret, buf, bufsize+1);
    // readlink is one of the rare functions that write out a C string, but does
never append a null to the output buffer(!)
    // stringToUTF8() always appends a null byte, so restore the character under
the null byte after the write.
}

```

```

    HEAP8[buf+len] = endChar;

    return len;
},doAccess:function(path, amode) {
    if (amode & ~7) {
        // need a valid mode
        return -28;
    }
    var lookup = FS.lookupPath(path, { follow: true });
    var node = lookup.node;
    if (!node) {
        return -44;
    }
    var perms = '';
    if (amode & 4) perms += 'r';
    if (amode & 2) perms += 'w';
    if (amode & 1) perms += 'x';
    if (perms /* otherwise, they've just passed F_OK */ && FS.nodePermissions(node,
    perms)) {
        return -2;
    }
    return 0;
},doReadv:function(stream, iov, iovcnt, offset) {
    var ret = 0;
    for (var i = 0; i < iovcnt; i++) {
        var ptr = HEAP32[((iov)+(i*8))>>2];
        var len = HEAP32[((iov)+(i*8 + 4))>>2];
        var curr = FS.read(stream, HEAP8,ptr, len, offset);
        if (curr < 0) return -1;
        ret += curr;
        if (curr < len) break; // nothing more to read
    }
    return ret;
},doWritev:function(stream, iov, iovcnt, offset) {
    var ret = 0;
    for (var i = 0; i < iovcnt; i++) {
        var ptr = HEAP32[((iov)+(i*8))>>2];
        var len = HEAP32[((iov)+(i*8 + 4))>>2];
        var curr = FS.write(stream, HEAP8,ptr, len, offset);
        if (curr < 0) return -1;
        ret += curr;
    }
    return ret;
},varargs:undefined,get:function() {
    assert(SYSCALLS.varargs != undefined);
    SYSCALLS.varargs += 4;
    var ret = HEAP32[((SYSCALLS.varargs)-(4))>>2];
    return ret;
},getStr:function(ptr) {
    var ret = UTF8ToString(ptr);
    return ret;
},getStreamFromFD:function(fd) {
    var stream = FS.getStream(fd);
    if (!stream) throw new FS.ErrnoError(8);
    return stream;
}

```

```

},get64:function(low, high) {
  if (low >= 0) assert(high === 0);
  else assert(high === -1);
  return low;
}};

function __syscall__newselect(nfds, readfds, writefds, exceptfds, timeout) {
try {

  // readfds are supported,
  // writefds checks socket open status
  // exceptfds not supported
  // timeout is always 0 - fully async
  assert(nfds <= 64, 'nfds must be less than or equal to 64'); // fd sets have 64
bits // TODO: this could be 1024 based on current musl headers
  assert(!exceptfds, 'exceptfds not supported');

  var total = 0;

  var srcReadLow = (readfds ? HEAP32[((readfds)>>2)] : 0),
      srcReadHigh = (readfds ? HEAP32[((readfds)+(4))>>2] : 0);
  var srcWriteLow = (writefds ? HEAP32[((writefds)>>2)] : 0),
      srcWriteHigh = (writefds ? HEAP32[((writefds)+(4))>>2] : 0);
  var srcExceptLow = (exceptfds ? HEAP32[((exceptfds)>>2)] : 0),
      srcExceptHigh = (exceptfds ? HEAP32[((exceptfds)+(4))>>2] : 0);

  var dstReadLow = 0,
      dstReadHigh = 0;
  var dstWriteLow = 0,
      dstWriteHigh = 0;
  var dstExceptLow = 0,
      dstExceptHigh = 0;

  var allLow = (readfds ? HEAP32[((readfds)>>2)] : 0) |
               (writefds ? HEAP32[((writefds)>>2)] : 0) |
               (exceptfds ? HEAP32[((exceptfds)>>2)] : 0);
  var allHigh = (readfds ? HEAP32[((readfds)+(4))>>2] : 0) |
                (writefds ? HEAP32[((writefds)+(4))>>2] : 0) |
                (exceptfds ? HEAP32[((exceptfds)+(4))>>2] : 0);

  var check = function(fd, low, high, val) {
    return (fd < 32 ? (low & val) : (high & val));
  };

  for (var fd = 0; fd < nfds; fd++) {
    var mask = 1 << (fd % 32);
    if (!(check(fd, allLow, allHigh, mask))) {
      continue; // index isn't in the set
    }

    var stream = FS.getStream(fd);
    if (!stream) throw new FS.ErrnoError(8);

    var flags = SYSCALLS.DEFAULT_POLLMASK;

    if (stream.stream_ops.poll) {

```

```

        flags = stream.stream_ops.poll(stream);
    }

    if ((flags & 1) && check(fd, srcReadLow, srcReadHigh, mask)) {
        fd < 32 ? (dstReadLow = dstReadLow | mask) : (dstReadHigh = dstReadHigh | mask);
        total++;
    }
    if ((flags & 4) && check(fd, srcWriteLow, srcWriteHigh, mask)) {
        fd < 32 ? (dstWriteLow = dstWriteLow | mask) : (dstWriteHigh = dstWriteHigh | mask);
        total++;
    }
    if ((flags & 2) && check(fd, srcExceptLow, srcExceptHigh, mask)) {
        fd < 32 ? (dstExceptLow = dstExceptLow | mask) : (dstExceptHigh = dstExceptHigh | mask);
        total++;
    }
}

if (readfds) {
    HEAP32[((readfds)>>2)] = dstReadLow;
    HEAP32[((readfds)+(4))>>2] = dstReadHigh;
}
if (writefds) {
    HEAP32[((writefds)>>2)] = dstWriteLow;
    HEAP32[((writefds)+(4))>>2] = dstWriteHigh;
}
if (exceptfds) {
    HEAP32[((exceptfds)>>2)] = dstExceptLow;
    HEAP32[((exceptfds)+(4))>>2] = dstExceptHigh;
}

return total;
} catch (e) {
if (typeof FS == 'undefined' || !(e instanceof FS.ErrnoError)) throw e;
return -e(errno);
}
}

var SOCKFS = {mount:function(mount) {
    // If Module['websocket'] has already been defined (e.g. for configuring
    // the subprotocol/url) use that, if not initialise it to a new object.
    Module['websocket'] = (Module['websocket'] &&
        ('object' === typeof Module['websocket'])) ?
Module['websocket'] : {};
    // Add the Event registration mechanism to the exported websocket configuration
    // object so we can register network callbacks from native JavaScript too.
    // For more documentation see system/include/emscripten/emscripten.h
    Module['websocket'].callbacks = {};
    Module['websocket']['on'] = /** @this{Object} */ function(event, callback) {
        if ('function' === typeof callback) {
            this.callbacks[event] = callback;
        }
    }
}
}

```

```

        return this;
    };

Module['websocket'].emit = /** @this{Object} */ function(event, param) {
    if ('function' === typeof this._callbacks[event]) {
        this._callbacks[event].call(this, param);
    }
};

// If debug is enabled register simple default logging callbacks for each
Event.

return FS.createNode(null, '/', 16384 | 511 /* 0777 */, 0);
},createSocket:function(family, type, protocol) {
    type &= ~526336; // Some applications may pass it; it makes no sense for a
single process.
    var streaming = type == 1;
    if (streaming && protocol && protocol != 6) {
        throw new FS.ErrnoError(66); // if SOCK_STREAM, must be tcp or 0.
    }

    // create our internal socket structure
    var sock = {
        family: family,
        type: type,
        protocol: protocol,
        server: null,
        error: null, // Used in getsockopt for SOL_SOCKET/SO_ERROR test
        peers: {},
        pending: [],
        recv_queue: [],
        sock_ops: SOCKFS.websocket_sock_ops
    };

    // create the filesystem node to store the socket structure
    var name = SOCKFS.nextname();
    var node = FS.createNode(SOCKFS.root, name, 49152, 0);
    node.sock = sock;

    // and the wrapping stream that enables library functions such
    // as read and write to indirectly interact with the socket
    var stream = FS.createStream({
        path: name,
        node: node,
        flags: 2,
        seekable: false,
        stream_ops: SOCKFS.stream_ops
    });

    // map the new stream to the socket structure (sockets have a 1:1
    // relationship with a stream)
    sock.stream = stream;

    return sock;
},getSocket:function(fd) {

```

```

var stream = FS.getStream(fd);
if (!stream || !FS.isSocket(stream.node.mode)) {
    return null;
}
return stream.node.sock;
},stream_ops:{poll:function(stream) {
    var sock = stream.node.sock;
    return sock.sock_ops.poll(sock);
},ioctl:function(stream, request, varargs) {
    var sock = stream.node.sock;
    return sock.sock_ops.ioctl(sock, request, varargs);
},read:function(stream, buffer, offset, length, position /* ignored */) {
    var sock = stream.node.sock;
    var msg = sock.sock_ops.recvmsg(sock, length);
    if (!msg) {
        // socket is closed
        return 0;
    }
    buffer.set(msg.buffer, offset);
    return msg.buffer.length;
},write:function(stream, buffer, offset, length, position /* ignored */) {
    var sock = stream.node.sock;
    return sock.sock_ops.sendmsg(sock, buffer, offset, length);
},close:function(stream) {
    var sock = stream.node.sock;
    sock.sock_ops.close(sock);
},nextname:function() {
if (!SOCKFS.nextname.current) {
    SOCKFS.nextname.current = 0;
}
return 'socket[' + (SOCKFS.nextname.current++) + ']';
},websocket_sock_ops:{createPeer:function(sock, addr, port) {
    var ws;

    if (typeof addr == 'object') {
        ws = addr;
        addr = null;
        port = null;
    }

    if (ws) {
        // for sockets that've already connected (e.g. we're the server)
        // we can inspect the _socket property for the address
        if (ws._socket) {
            addr = ws._socket.remoteAddress;
            port = ws._socket.remotePort;
        }
        // if we're just now initializing a connection to the remote,
        // inspect the url property
        else {
            var result = /ws[s]?:\/\/([^\:]+):(\d+)/.exec(ws.url);
            if (!result) {
                throw new Error('WebSocket URL must be in the format
ws(s)://address:port');
            }
        }
    }
}

```

```

        addr = result[1];
        port = parseInt(result[2], 10);
    }
} else {
    // create the actual websocket object and connect
    try {
        // runtimeConfig gets set to true if WebSocket runtime configuration is
available.
        var runtimeConfig = (Module['websocket'] && ('object' === typeof
Module['websocket']));
    }

        // The default value is 'ws://' the replace is needed because the
compiler replaces '//' comments with '#'
        // comments without checking context, so we'd end up with ws:#, the
replace swaps the '#' for '//' again.
        var url = 'ws:#'.replace('#', '//');

        if (runtimeConfig) {
            if ('string' === typeof Module['websocket']['url']) {
                url = Module['websocket']['url']; // Fetch runtime WebSocket URL
config.
            }
        }

        if (url === 'ws://' || url === 'wss://') { // Is the supplied URL config
just a prefix, if so complete it.
            var parts = addr.split('/');
            url = url + parts[0] + ":" + port + "/" + parts.slice(1).join('/');
        }

        // Make the WebSocket subprotocol (Sec-WebSocket-Protocol) default to
binary if no configuration is set.
        var subProtocols = 'binary'; // The default value is 'binary'

        if (runtimeConfig) {
            if ('string' === typeof Module['websocket']['subprotocol']) {
                subProtocols = Module['websocket']['subprotocol']; // Fetch runtime
WebSocket subprotocol config.
            }
        }

        // The default WebSocket options
        var opts = undefined;

        if (subProtocols !== 'null') {
            // The regex trims the string (removes spaces at the beginning and end,
then splits the string by
            // <any space>,<any space> into an Array. Whitespace removal is
important for Websockify and ws.
            subProtocols = subProtocols.replace(/^\s+|\s+$/g,"").split(/\s*,\s*/);

            // The node ws library API for specifying optional subprotocol is
slightly different than the browser's.
            opts = ENVIRONMENT_IS_NODE ? {'protocol': subProtocols.toString()} :
subProtocols;
        }
    }
}

```

```

    }

    // some webservers (azure) does not support subprotocol header
    if (runtimeConfig && null === Module['websocket']['subprotocol']) {
        subProtocols = 'null';
        opts = undefined;
    }

    // If node we use the ws library.
    var WebSocketConstructor;
    if (ENVIRONMENT_IS_NODE) {
        WebSocketConstructor = /** @type{(typeof WebSocket)} */(require('ws'));
    } else {
    }
    WebSocketConstructor = WebSocket;
}

ws = new WebSocketConstructor(url, opts);
ws.binaryType = 'arraybuffer';
} catch (e) {
    throw new FS.ErrnoError(23);
}
}

var peer = {
    addr: addr,
    port: port,
    socket: ws,
    dgram_send_queue: []
};

SOCKFS.websocket_sock_ops.addPeer(sock, peer);
SOCKFS.websocket_sock_ops.handlePeerEvents(sock, peer);

// if this is a bound dgram socket, send the port number first to allow
// us to override the ephemeral port reported to us by remotePort on the
// remote end.
if (sock.type === 2 && typeof sock.sport != 'undefined') {
    peer.dgram_send_queue.push(new Uint8Array([
        255, 255, 255, 255,
        'p'.charCodeAt(0), 'o'.charCodeAt(0), 'r'.charCodeAt(0),
        't'.charCodeAt(0),
        ((sock.sport & 0xff00) >> 8), (sock.sport & 0xff)
    ]));
}

return peer;
},getPeer:function(sock, addr, port) {
    return sock.peers[addr + ':' + port];
},addPeer:function(sock, peer) {
    sock.peers[peer.addr + ':' + peer.port] = peer;
},removePeer:function(sock, peer) {
    delete sock.peers[peer.addr + ':' + peer.port];
},handlePeerEvents:function(sock, peer) {
    var first = true;
}
}

```

```

var handleOpen = function () {
    Module['websocket'].emit('open', sock.stream.fd);

    try {
        var queued = peer.dgram_send_queue.shift();
        while (queued) {
            peer.socket.send(queued);
            queued = peer.dgram_send_queue.shift();
        }
    } catch (e) {
        // not much we can do here in the way of proper error handling as we've
already
        // lied and said this data was sent. shut it down.
        peer.socket.close();
    }
};

function handleMessage(data) {
    if (typeof data == 'string') {
        var encoder = new TextEncoder(); // should be utf-8
        data = encoder.encode(data); // make a typed array from the string
    } else {
        assert(data.byteLength !== undefined); // must receive an ArrayBuffer
        if (data.byteLength == 0) {
            // An empty ArrayBuffer will emit a pseudo disconnect event
            // as recv/recvmsg will return zero which indicates that a socket
            // has performed a shutdown although the connection has not been
disconnected yet.
            return;
        } else {
            data = new Uint8Array(data); // make a typed array view on the array
buffer
        }
    }

    // if this is the port message, override the peer's port with it
    var wasfirst = first;
    first = false;
    if (wasfirst &&
        data.length === 10 &&
        data[0] === 255 && data[1] === 255 && data[2] === 255 && data[3] ===
255 &&
        data[4] === 'p'.charCodeAt(0) && data[5] === 'o'.charCodeAt(0) &&
data[6] === 'r'.charCodeAt(0) && data[7] === 't'.charCodeAt(0)) {
        // update the peer's port and it's key in the peer map
        var newport = ((data[8] << 8) | data[9]);
        SOCKFS.websocket_sock_ops.removePeer(sock, peer);
        peer.port = newport;
        SOCKFS.websocket_sock_ops.addPeer(sock, peer);
        return;
    }

    sock.recv_queue.push({ addr: peer.addr, port: peer.port, data: data });
    Module['websocket'].emit('message', sock.stream.fd);
}

```

```

};

if (ENVIRONMENT_IS_NODE) {
  peer.socket.on('open', handleOpen);
  peer.socket.on('message', function(data, flags) {
    if (!flags.binary) {
      return;
    }
    handleMessage((new Uint8Array(data)).buffer); // copy from node Buffer -
> ArrayBuffer
  });
  peer.socket.on('close', function() {
    Module['websocket'].emit('close', sock.stream.fd);
  });
  peer.socket.on('error', function(error) {
    // Although the ws library may pass errors that may be more descriptive
than
    // ECONNREFUSED they are not necessarily the expected error code e.g.
    // ENOTFOUND on getaddrinfo seems to be node.js specific, so using
ECONNREFUSED
    // is still probably the most useful thing to do.
    sock.error = 14; // Used in getsockopt for SOL_SOCKET/SO_ERROR test.
    Module['websocket'].emit('error', [sock.stream.fd, sock.error,
'ECONNREFUSED: Connection refused']);
    // don't throw
  });
} else {
  peer.socket.onopen = handleOpen;
  peer.socket.onclose = function() {
    Module['websocket'].emit('close', sock.stream.fd);
  };
  peer.socket.onmessage = function peer_socket_onmessage(event) {
    handleMessage(event.data);
  };
  peer.socket.onerror = function(error) {
    // The WebSocket spec only allows a 'simple event' to be thrown on error,
    // so we only really know as much as ECONNREFUSED.
    sock.error = 14; // Used in getsockopt for SOL_SOCKET/SO_ERROR test.
    Module['websocket'].emit('error', [sock.stream.fd, sock.error,
'ECONNREFUSED: Connection refused']);
  };
}
},poll:function(sock) {
  if (sock.type === 1 && sock.server) {
    // listen sockets should only say they're available for reading
    // if there are pending clients.
    return sock.pending.length ? (64 | 1) : 0;
  }

  var mask = 0;
  var dest = sock.type === 1 ? // we only care about the socket state for
connection-based sockets
    SOCKFS.websocket_sock_ops.getPeer(sock, sock.daddr, sock.dport) :
  null;
}

```

```

    if (sock.recv_queue.length ||
        !dest || // connection-less sockets are always ready to read
        (dest && dest.socket.readyState === dest.socket.CLOSING) ||
        (dest && dest.socket.readyState === dest.socket.CLOSED)) { // let recv
return 0 once closed
    mask |= (64 | 1);
}

if (!dest || // connection-less sockets are always ready to write
    (dest && dest.socket.readyState === dest.socket.OPEN)) {
    mask |= 4;
}

if ((dest && dest.socket.readyState === dest.socket.CLOSING) ||
    (dest && dest.socket.readyState === dest.socket.CLOSED)) {
    mask |= 16;
}

return mask;
},ioctl:function(sock, request, arg) {
    switch (request) {
        case 21531:
            var bytes = 0;
            if (sock.recv_queue.length) {
                bytes = sock.recv_queue[0].data.length;
            }
            HEAP32[((arg)>>2)] = bytes;
            return 0;
        default:
            return 28;
    }
},close:function(sock) {
// if we've spawned a listen server, close it
if (sock.server) {
    try {
        sock.server.close();
    } catch (e) {
    }
    sock.server = null;
}
// close any peer connections
var peers = Object.keys(sock.peers);
for (var i = 0; i < peers.length; i++) {
    var peer = sock.peers[peers[i]];
    try {
        peer.socket.close();
    } catch (e) {
    }
    SOCKFS.websocket_sock_ops.removePeer(sock, peer);
}
return 0;
},bind:function(sock, addr, port) {
    if (typeof sock.saddr != 'undefined' || typeof sock.sport != 'undefined') {
        throw new FS.ErrnoError(28); // already bound
    }
}

```

```

    sock.saddr = addr;
    sock.sport = port;
    // in order to emulate dgram sockets, we need to launch a listen server when
    // binding on a connection-less socket
    // note: this is only required on the server side
    if (sock.type === 2) {
        // close the existing server if it exists
        if (sock.server) {
            sock.server.close();
            sock.server = null;
        }
        // swallow error operation not supported error that occurs when binding in
the
        // browser where this isn't supported
        try {
            sock.sock_ops.listen(sock, 0);
        } catch (e) {
            if (!(e instanceof FS.ErrnoError)) throw e;
            if (e.errno !== 138) throw e;
        }
    }
},connect:function(sock, addr, port) {
    if (sock.server) {
        throw new FS.ErrnoError(138);
    }

    // TODO autobind
    // if (!sock.addr && sock.type == 2) {
    // }

    // early out if we're already connected / in the middle of connecting
    if (typeof sock.daddr != 'undefined' && typeof sock.dport != 'undefined') {
        var dest = SOCKFS.websocket_sock_ops.getPeer(sock, sock.daddr, sock.dport);
        if (dest) {
            if (dest.socket.readyState === dest.socket.CONNECTING) {
                throw new FS.ErrnoError(7);
            } else {
                throw new FS.ErrnoError(30);
            }
        }
    }
}

// add the socket to our peer list and set our
// destination address / port to match
var peer = SOCKFS.websocket_sock_ops.createPeer(sock, addr, port);
sock.daddr = peer.addr;
sock.dport = peer.port;

// always "fail" in non-blocking mode
throw new FS.ErrnoError(26);
},listen:function(sock, backlog) {
    if (!ENVIRONMENT_IS_NODE) {
        throw new FS.ErrnoError(138);
    }
    if (sock.server) {

```

```

        throw new FS.ErrnoError(28); // already listening
    }
    var WebSocketServer = require('ws').Server;
    var host = sock.saddr;
    sock.server = new WebSocketServer({
        host: host,
        port: sock.sport
        // TODO support backlog
    });
    Module['websocket'].emit('listen', sock.stream.fd); // Send Event with listen
fd.

    sock.server.on('connection', function(ws) {
        if (sock.type === 1) {
            var newsock = SOCKFS.createSocket(sock.family, sock.type, sock.protocol);

            // create a peer on the new socket
            var peer = SOCKFS.websocket_sock_ops.createPeer(newsock, ws);
            newsock.daddr = peer.addr;
            newsock.dport = peer.port;

            // push to queue for accept to pick up
            sock.pending.push(newsock);
            Module['websocket'].emit('connection', newsock.stream.fd);
        } else {
            // create a peer on the listen socket so calling sendto
            // with the listen socket and an address will resolve
            // to the correct client
            SOCKFS.websocket_sock_ops.createPeer(sock, ws);
            Module['websocket'].emit('connection', sock.stream.fd);
        }
    });
    sock.server.on('closed', function() {
        Module['websocket'].emit('close', sock.stream.fd);
        sock.server = null;
    });
    sock.server.on('error', function(error) {
        // Although the ws library may pass errors that may be more descriptive
than
        // ECONNREFUSED they are not necessarily the expected error code e.g.
        // ENOTFOUND on getaddrinfo seems to be node.js specific, so using
EHOSTUNREACH
        // is still probably the most useful thing to do. This error shouldn't
        // occur in a well written app as errors should get trapped in the compiled
        // app's own getaddrinfo call.
        sock.error = 23; // Used in getsockopt for SOL_SOCKET/SO_ERROR test.
        Module['websocket'].emit('error', [sock.stream.fd, sock.error,
'EHOSTUNREACH: Host is unreachable']);
        // don't throw
    });
},accept:function(listensock) {
    if (!listensock.server || !listensock.pending.length) {
        throw new FS.ErrnoError(28);
    }
    var newsock = listensock.pending.shift();
}

```

```

    newsock.stream.flags = listensock.stream.flags;
    return newsock;
}, getname: function(sock, peer) {
    var addr, port;
    if (peer) {
        if (sock.daddr === undefined || sock.dport === undefined) {
            throw new FS.ErrnoError(53);
        }
        addr = sock.daddr;
        port = sock.dport;
    } else {
        // TODO saddr and sport will be set for bind()'d UDP sockets, but what
        // should we be returning for TCP sockets that've been connect()'d?
        addr = sock.saddr || 0;
        port = sock.sport || 0;
    }
    return { addr: addr, port: port };
}, sendmsg: function(sock, buffer, offset, length, addr, port) {
    if (sock.type === 2) {
        // connection-less sockets will honor the message address,
        // and otherwise fall back to the bound destination address
        if (addr === undefined || port === undefined) {
            addr = sock.daddr;
            port = sock.dport;
        }
        // if there was no address to fall back to, error out
        if (addr === undefined || port === undefined) {
            throw new FS.ErrnoError(17);
        }
    } else {
        // connection-based sockets will only use the bound
        addr = sock.daddr;
        port = sock.dport;
    }

    // find the peer for the destination address
    var dest = SOCKFS.websocket_sock_ops.getPeer(sock, addr, port);

    // early out if not connected with a connection-based socket
    if (sock.type === 1) {
        if (!dest || dest.socket.readyState === dest.socket.CLOSING ||
dest.socket.readyState === dest.socket.CLOSED) {
            throw new FS.ErrnoError(53);
        } else if (dest.socket.readyState === dest.socket.CONNECTING) {
            throw new FS.ErrnoError(6);
        }
    }

    // create a copy of the incoming data to send, as the WebSocket API
    // doesn't work entirely with an ArrayBufferView, it'll just send
    // the entire underlying buffer
    if (ArrayBuffer.isView(buffer)) {
        offset += buffer.byteOffset;
        buffer = buffer.buffer;
    }
}

```

```

var data;
data = buffer.slice(offset, offset + length);

// if we're emulating a connection-less dgram socket and don't have
// a cached connection, queue the buffer to send upon connect and
// lie, saying the data was sent now.
if (sock.type === 2) {
    if (!dest || dest.socket.readyState !== dest.socket.OPEN) {
        // if we're not connected, open a new connection
        if (!dest || dest.socket.readyState === dest.socket.CLOSING ||
dest.socket.readyState === dest.socket.CLOSED) {
            dest = SOCKFS.websocket_sock_ops.createPeer(sock, addr, port);
        }
        dest.dgram_send_queue.push(data);
        return length;
    }
}

try {
    // send the actual data
    dest.socket.send(data);
    return length;
} catch (e) {
    throw new FS.ErrnoError(28);
}
},recvmsg:function(sock, length) {
// http://pubs.opengroup.org/onlinepubs/7908799/xns/recvmsg.html
if (sock.type === 1 && sock.server) {
    // tcp servers should not be recv()'ing on the listen socket
    throw new FS.ErrnoError(53);
}

var queued = sock.recv_queue.shift();
if (!queued) {
    if (sock.type === 1) {
        var dest = SOCKFS.websocket_sock_ops.getPeer(sock, sock.daddr,
sock.dport);

        if (!dest) {
            // if we have a destination address but are not connected, error out
            throw new FS.ErrnoError(53);
        }
        else if (dest.socket.readyState === dest.socket.CLOSING ||
dest.socket.readyState === dest.socket.CLOSED) {
            // return null if the socket has closed
            return null;
        }
        else {
            // else, our socket is in a valid state but truly has nothing available
            throw new FS.ErrnoError(6);
        }
    } else {
        throw new FS.ErrnoError(6);
    }
}
}

```

```

    }

    // queued.data will be an ArrayBuffer if it's unadulterated, but if it's
    // requeued TCP data it'll be an ArrayBufferView
    var queuedLength = queued.data.byteLength || queued.data.length;
    var queuedOffset = queued.data.byteOffset || 0;
    var queuedBuffer = queued.data.buffer || queued.data;
    var bytesRead = Math.min(length, queuedLength);
    var res = {
        buffer: new Uint8Array(queuedBuffer, queuedOffset, bytesRead),
        addr: queued.addr,
        port: queued.port
    };

    // push back any unread data for TCP connections
    if (sock.type === 1 && bytesRead < queuedLength) {
        var bytesRemaining = queuedLength - bytesRead;
        queued.data = new Uint8Array(queuedBuffer, queuedOffset + bytesRead,
bytesRemaining);
        sock.recv_queue.unshift(queued);
    }
}

return res;
}};

function getSocketFromFD(fd) {
    var socket = SOCKFS.getSocket(fd);
    if (!socket) throw new FS.ErrnoError(8);
    return socket;
}

function setErrNo(value) {
    HEAP32[((__errno_location())>>2)] = value;
    return value;
}

var Sockets =
{BUFFER_SIZE:10240,MAX_BUFFER_SIZE:10485760,nextFd:1,fds:{},nextport:1,maxport:65535,pe
er:null,connections:{},portmap:{},localAddr:4261412874,addrPool:[33554442,50331658,6710
8874,83886090,100663306,117440522,134217738,150994954,167772170,184549386,201326602,218
103818,234881034]};

function inetPton4(str) {
    var b = str.split('.');
    for (var i = 0; i < 4; i++) {
        var tmp = Number(b[i]);
        if (isNaN(tmp)) return null;
        b[i] = tmp;
    }
    return (b[0] | (b[1] << 8) | (b[2] << 16) | (b[3] << 24)) >>> 0;
}

/** @suppress {checkTypes} */
function atoi_q(str) {
    return parseInt(str);
}

function inetPton6(str) {
    var words;

```

```

var w, offset, z, i;
/* http://home.deds.nl/~aeron/regex/ */
var valid6regex = /^(=(?:.*::)(?!.*::.+::)(::)?([\dA-F]{1,4}:(|[\b])|){5}|([\dA-
F]{1,4}::){6})((([\dA-F]{1,4}((?!\\3)::|:\\b|$))|(?!\2\\3)){2}|(((2[0-4]|1\\d|[1-
9])?\\d|25[0-5])\\.?\\b){4}))$/i
var parts = [];
if (!valid6regex.test(str)) {
    return null;
}
if (str === "::") {
    return [0, 0, 0, 0, 0, 0, 0, 0];
}
// Z placeholder to keep track of zeros when splitting the string on ":" 
if (str.startsWith("::")) {
    str = str.replace("::", "Z:");
} else {
    str = str.replace("::", ":Z:");
}

if (str.indexOf(".") > 0) {
    // parse IPv4 embedded stress
    str = str.replace(new RegExp('[.]', 'g'), ":");

    words = str.split(":");
    words[words.length-4] = jstoi_q(words[words.length-4]) +
jstoi_q(words[words.length-3])*256;
    words[words.length-3] = jstoi_q(words[words.length-2]) +
jstoi_q(words[words.length-1])*256;
    words = words.slice(0, words.length-2);
} else {
    words = str.split(":");
}

offset = 0; z = 0;
for (w=0; w < words.length; w++) {
    if (typeof words[w] == 'string') {
        if (words[w] === 'Z') {
            // compressed zeros - write appropriate number of zero words
            for (z = 0; z < (8 - words.length+1); z++) {
                parts[w+z] = 0;
            }
            offset = z-1;
        } else {
            // parse hex to field to 16-bit value and write it in network byte-order
            parts[w+offset] = _htons(parseInt(words[w],16));
        }
    } else {
        // parsed IPv4 words
        parts[w+offset] = words[w];
    }
}
return [
    (parts[1] << 16) | parts[0],
    (parts[3] << 16) | parts[2],
    (parts[5] << 16) | parts[4],
    (parts[7] << 16) | parts[6]
]

```

```

        ];
    }
/** @param {number=} addrlen */
function writeSockaddr(sa, family, addr, port, addrlen) {
    switch (family) {
        case 2:
            addr = inetPton4(addr);
            zeroMemory(sa, 16);
            if (addrlen) {
                HEAP32[((addrlen)>>2)] = 16;
            }
            HEAP16[((sa)>>1)] = family;
            HEAP32[((((sa)+(4))>>2))] = addr;
            HEAP16[((((sa)+(2))>>1))] = _htons(port);
            break;
        case 10:
            addr = inetPton6(addr);
            zeroMemory(sa, 28);
            if (addrlen) {
                HEAP32[((addrlen)>>2)] = 28;
            }
            HEAP32[((sa)>>2)] = family;
            HEAP32[((((sa)+(8))>>2))] = addr[0];
            HEAP32[((((sa)+(12))>>2))] = addr[1];
            HEAP32[((((sa)+(16))>>2))] = addr[2];
            HEAP32[((((sa)+(20))>>2))] = addr[3];
            HEAP16[((((sa)+(2))>>1))] = _htons(port);
            break;
        default:
            return 5;
    }
    return 0;
}

var DNS = {address_map:{id:1,addrs:{},names:{}},lookup_name:function (name) {
    // If the name is already a valid ipv4 / ipv6 address, don't generate a fake
one.
    var res = inetPton4(name);
    if (res !== null) {
        return name;
    }
    res = inetPton6(name);
    if (res !== null) {
        return name;
    }

    // See if this name is already mapped.
    var addr;

    if (DNS.address_map.addrs[name]) {
        addr = DNS.address_map.addrs[name];
    } else {
        var id = DNS.address_map.id++;
        assert(id < 65535, 'exceeded max address mappings of 65535');
    }
}
}

```

```

    addr = '172.29.' + (id & 0xff) + '.' + (id & 0xff00);

    DNS.address_map.names[addr] = name;
    DNS.address_map.addrs[name] = addr;
}

return addr;
}, lookup_addr:function (addr) {
if (DNS.address_map.names[addr]) {
    return DNS.address_map.names[addr];
}

return null;
}};

function __syscall_accept4(fd, addr, addrlen, flags) {
try {

var sock = getSocketFromFD(fd);
var newsock = sock.sock_ops.accept(sock);
if (addr) {
    var errno = writeSockaddr(addr, newsock.family, DNS.lookup_name(newsock.daddr),
newsock.dport, addrlen);
    assert(!errno);
}
return newsock.stream.fd;
} catch (e) {
if (typeof FS == 'undefined' || !(e instanceof FS.ErrnoError)) throw e;
return -e(errno);
}
}

function __syscall_chmod(path, mode) {
try {

path = SYSCALLS.getStr(path);
FS.chmod(path, mode);
return 0;
} catch (e) {
if (typeof FS == 'undefined' || !(e instanceof FS.ErrnoError)) throw e;
return -e(errno);
}
}

function inetNtop4(addr) {
    return (addr & 0xff) + '.' + ((addr >> 8) & 0xff) + '.' + ((addr >> 16) & 0xff) +
'.' + ((addr >> 24) & 0xff)
}

function inetNtop6(ints) {
// ref: http://www.ietf.org/rfc/rfc2373.txt - section 2.5.4
// Format for IPv4 compatible and mapped 128-bit IPv6 Addresses
// 128-bits are split into eight 16-bit words
// stored in network byte order (big-endian)
// | 80 bits           | 16 | 32 bits           |
// +-----+-----+-----+

```

```

// | 10 bytes | 2 | 4 bytes |
// +-----+-----+-----+
// + 5 words | 1 | 2 words |
// +-----+-----+-----+
// |0000.....0000|0000| IPv4 ADDRESS |
(compatible)
// +-----+-----+-----+
// |0000.....0000|FFFF| IPv4 ADDRESS | (mapped)
// +-----+-----+-----+
var str = "";
var word = 0;
var longest = 0;
var lastzero = 0;
var zstart = 0;
var len = 0;
var i = 0;
var parts = [
    ints[0] & 0xffff,
    (ints[0] >> 16),
    ints[1] & 0xffff,
    (ints[1] >> 16),
    ints[2] & 0xffff,
    (ints[2] >> 16),
    ints[3] & 0xffff,
    (ints[3] >> 16)
];
// Handle IPv4-compatible, IPv4-mapped, loopback and any/unspecified addresses

var hasipv4 = true;
var v4part = "";
// check if the 10 high-order bytes are all zeros (first 5 words)
for (i = 0; i < 5; i++) {
    if (parts[i] !== 0) { hasipv4 = false; break; }
}

if (hasipv4) {
    // low-order 32-bits store an IPv4 address (bytes 13 to 16) (last 2 words)
    v4part = inetNtop4(parts[6] | (parts[7] << 16));
    // IPv4-mapped IPv6 address if 16-bit value (bytes 11 and 12) == 0xFFFF (6th
word)
    if (parts[5] === -1) {
        str = "::ffff:";
        str += v4part;
        return str;
    }
    // IPv4-compatible IPv6 address if 16-bit value (bytes 11 and 12) == 0x0000
(6th word)
    if (parts[5] === 0) {
        str = "::";
        //special case IPv6 addresses
        if (v4part === "0.0.0.0") v4part = ""; // any/unspecified address
        if (v4part === "0.0.0.1") v4part = "1"; // loopback address
        str += v4part;
        return str;
    }
}

```

```

        }
    }

    // Handle all other IPv6 addresses

    // first run to find the longest contiguous zero words
    for (word = 0; word < 8; word++) {
        if (parts[word] === 0) {
            if (word - lastzero > 1) {
                len = 0;
            }
            lastzero = word;
            len++;
        }
        if (len > longest) {
            longest = len;
            zstart = word - longest + 1;
        }
    }

    for (word = 0; word < 8; word++) {
        if (longest > 1) {
            // compress contiguous zeros - to produce "::"
            if (parts[word] === 0 && word >= zstart && word < (zstart + longest) ) {
                if (word === zstart) {
                    str += ":";  

                    if (zstart === 0) str += ":"; //leading zeros case
                }
                continue;
            }
            // converts 16-bit words from big-endian to little-endian before converting to
            hex string
            str += Number(_ntohs(parts[word] & 0xffff)).toString(16);
            str += word < 7 ? ":" : "";
        }
        return str;
    }
}

function readSockaddr(sa, salen) {
    // family / port offsets are common to both sockaddr_in and sockaddr_in6
    var family = HEAP16[((sa)>>1)];
    var port = _ntohs(HEAPU16[((sa)+(2))>>1]);
    var addr;

    switch (family) {
        case 2:
            if (salen !== 16) {
                return { errno: 28 };
            }
            addr = HEAP32[((sa)+(4))>>2];
            addr = inetNtop4(addr);
            break;
        case 10:
            if (salen !== 28) {
                return { errno: 28 };
            }
    }
}

```

```

        }
        addr = [
            HEAP32[((sa)+(8))>>2],
            HEAP32[((sa)+(12))>>2],
            HEAP32[((sa)+(16))>>2],
            HEAP32[((sa)+(20))>>2]
        ];
        addr = inetNtop6(addr);
        break;
    default:
        return { errno: 5 };
    }

    return { family: family, addr: addr, port: port };
}
/** @param {boolean=} allowNull */
function getSocketAddress(addrp, addrlen, allowNull) {
    if (allowNull && addrp === 0) return null;
    var info = readSockaddr(addrp, addrlen);
    if (info.errno) throw new FS.ErrnoError(info(errno));
    info.addr = DNS.lookup_addr(info.addr) || info.addr;
    return info;
}
function __syscall_connect(fd, addr, addrlen) {
try {

    var sock = getSocketFromFD(fd);
    var info = getSocketAddress(addr, addrlen);
    sock.sock_ops.connect(sock, info.addr, info.port);
    return 0;
} catch (e) {
    if (typeof FS == 'undefined' || !(e instanceof FS.ErrnoError)) throw e;
    return -e.errno;
}
}

function __syscall_faccessat(dirfd, path, amode, flags) {
try {

    path = SYSCALLS.getPath(path);
    assert(flags === 0);
    path = SYSCALLS.calculateAt(dirfd, path);
    return SYSCALLS.doAccess(path, amode);
} catch (e) {
    if (typeof FS == 'undefined' || !(e instanceof FS.ErrnoError)) throw e;
    return -e.errno;
}
}

function __syscall_fcntl64(fd, cmd, varargs) {
SYSCALLS.varargs = varargs;
try {

    var stream = SYSCALLS.getStreamFromFD(fd);
    switch (cmd) {

```

```

case 0: {
    var arg = SYSCALLS.get();
    if (arg < 0) {
        return -28;
    }
    var newStream;
    newStream = FS.open(stream.path, stream.flags, 0, arg);
    return newStream.fd;
}
case 1:
case 2:
    return 0; // FD_CLOEXEC makes no sense for a single process.
case 3:
    return stream.flags;
case 4: {
    var arg = SYSCALLS.get();
    stream.flags |= arg;
    return 0;
}
case 5:
/* case 5: Currently in musl F_GETLK64 has same value as F_GETLK, so omitted to
   avoid duplicate case blocks. If that changes, uncomment this */
    var arg = SYSCALLS.get();
    var offset = 0;
    // We're always unlocked.
    HEAP16[((arg)+(offset))>>1] = 2;
    return 0;
}
case 6:
case 7:
/* case 6: Currently in musl F_SETLK64 has same value as F_SETLK, so omitted to
   avoid duplicate case blocks. If that changes, uncomment this */
/* case 7: Currently in musl F_SETLKW64 has same value as F_SETLKW, so omitted
   to avoid duplicate case blocks. If that changes, uncomment this */

    return 0; // Pretend that the locking is successful.
case 16:
case 8:
    return -28; // These are for sockets. We don't have them fully implemented
yet.
case 9:
    // musl trusts getown return values, due to a bug where they must be, as they
overlap with errors. just return -1 here, so fnctl() returns that, and we set errno
ourselves.
    setErrNo(28);
    return -1;
default: {
    return -28;
}
}
} catch (e) {
if (typeof FS == 'undefined' || !(e instanceof FS.ErrnoError)) throw e;
return -e(errno);
}

```

```

}

}

function __syscall_fstat64(fd, buf) {
try {

    var stream = SYSCALLS.getStreamFromFD(fd);
    return SYSCALLS.doStat(FS.stat, stream.path, buf);
} catch (e) {
if (typeof FS == 'undefined' || !(e instanceof FS.ErrnoError)) throw e;
    return -e(errno);
}
}

function __syscall_getcwd(buf, size) {
try {

    if (size === 0) return -28;
    var cwd = FS.cwd();
    var cwdLengthInBytes = lengthBytesUTF8(cwd);
    if (size < cwdLengthInBytes + 1) return -68;
    stringToUTF8(cwd, buf, size);
    return buf;
} catch (e) {
if (typeof FS == 'undefined' || !(e instanceof FS.ErrnoError)) throw e;
    return -e(errno);
}
}

function __syscall_getdents64(fd, dirp, count) {
try {

    var stream = SYSCALLS.getStreamFromFD(fd)
    if (!stream.getdents) {
        stream.getdents = FS.readdir(stream.path);
    }

    var struct_size = 280;
    var pos = 0;
    var off = FS.llseek(stream, 0, 1);

    var idx = Math.floor(off / struct_size);

    while (idx < stream.getdents.length && pos + struct_size <= count) {
        var id;
        var type;
        var name = stream.getdents[idx];
        if (name === '.') {
            id = stream.node.id;
            type = 4; // DT_DIR
        }
        else if (name === '..') {
            var lookup = FS.lookupPath(stream.path, { parent: true });
            id = lookup.node.id;
            type = 4; // DT_DIR
        }
    }
}
}

```

```

    }
    else {
        var child = FS.lookupNode(stream.node, name);
        id = child.id;
        type = FS.isChrdev(child.mode) ? 2 : // DT_CHR, character device.
            FS.isDir(child.mode) ? 4 : // DT_DIR, directory.
            FS.isLink(child.mode) ? 10 : // DT_LNK, symbolic link.
            8; // DT_REG, regular file.
    }
    assert(id);
    (tempI64 = [id>>>0,(tempDouble=id, (+(Math.abs(tempDouble))) >= 1.0 ?
        (tempDouble > 0.0 ? ((Math.min((+(Math.floor((tempDouble)/4294967296.0))),4294967295.0))|0)>>>0 : (~(~((+(Math.ceil((tempDouble -
        +((~(tempDouble))>>>0))/4294967296.0))))>>>0) : 0)],HEAP32[((dirp + pos)>>2)] =
        tempI64[0],HEAP32[((dirp + pos)+(4))>>2]) = tempI64[1]);
    (tempI64 = [(idx + 1) * struct_size>>>0,(tempDouble=(idx + 1) *
        struct_size,(+Math.abs(tempDouble))) >= 1.0 ? (tempDouble > 0.0 ?
        ((Math.min((+(Math.floor((tempDouble)/4294967296.0))),4294967295.0))|0)>>>0 :
        (~(~((+(Math.ceil((tempDouble - +((~(tempDouble))>>>0))/4294967296.0))))>>>0) : 0)],HEAP32[((dirp + pos)+(8))>>2]) = tempI64[0],HEAP32[((dirp + pos)+(12))>>2]) =
        tempI64[1]);
    HEAP16[((dirp + pos)+(16))>>1] = 280;
    HEAP8[((dirp + pos)+(18))>>0] = type;
    stringToUTF8(name, dirp + pos + 19, 256);
    pos += struct_size;
    idx += 1;
}
FS.llseek(stream, idx * struct_size, 0);
return pos;
} catch (e) {
if (typeof FS == 'undefined' || !(e instanceof FS.ErrnoError)) throw e;
return -e(errno);
}
}

function __syscall_getsockopt(fd, level, optname, optval, optlen) {
try {

    var sock = getSocketFromFD(fd);
    // Minimal getsockopt aimed at resolving https://github.com/emscripten-
    core/emscripten/issues/2211
    // so only supports SOL_SOCKET with SO_ERROR.
    if (level === 1) {
        if (optname === 4) {
            HEAP32[((optval)>>2)] = sock.error;
            HEAP32[((optlen)>>2)] = 4;
            sock.error = null; // Clear the error (The SO_ERROR option obtains and then
            clears this field).
            return 0;
        }
    }
    return -50; // The option is unknown at the level indicated.
} catch (e) {
if (typeof FS == 'undefined' || !(e instanceof FS.ErrnoError)) throw e;
return -e(errno);
}
}

```

```
}

function __syscall_ioctl(fd, op, varargs) {
SYSCALLS.varargs = varargs;
try {

    var stream = SYSCALLS.getStreamFromFD(fd);
    switch (op) {
        case 21509:
        case 21505: {
            if (!stream.tty) return -59;
            return 0;
        }
        case 21510:
        case 21511:
        case 21512:
        case 21506:
        case 21507:
        case 21508: {
            if (!stream.tty) return -59;
            return 0; // no-op, not actually adjusting terminal settings
        }
        case 21519: {
            if (!stream.tty) return -59;
            var argp = SYSCALLS.get();
            HEAP32[((argp)>>2)] = 0;
            return 0;
        }
        case 21520: {
            if (!stream.tty) return -59;
            return -28; // not supported
        }
        case 21531: {
            var argp = SYSCALLS.get();
            return FS.ioctl(stream, op, argp);
        }
        case 21523: {
            // TODO: in theory we should write to the winsize struct that gets
            // passed in, but for now musl doesn't read anything on it
            if (!stream.tty) return -59;
            return 0;
        }
        case 21524: {
            // TODO: technically, this ioctl call should change the window size.
            // but, since emscripten doesn't have any concept of a terminal window
            // yet, we'll just silently throw it away as we do TIOCGWINSZ
            if (!stream.tty) return -59;
            return 0;
        }
        default: abort('bad ioctl syscall ' + op);
    }
} catch (e) {
if (typeof FS == 'undefined' || !(e instanceof FS.ErrnoError)) throw e;
return -e(errno);
}
}
```

```

}

function __syscall_lstat64(path, buf) {
try {

    path = SYSCALLS.getStr(path);
    return SYSCALLS.doStat(FS.lstat, path, buf);
} catch (e) {
if (typeof FS == 'undefined' || !(e instanceof FS.ErrnoError)) throw e;
return -e(errno);
}
}

function __syscall_mkdir(path, mode) {
try {

    path = SYSCALLS.getStr(path);
    return SYSCALLS.doMkdir(path, mode);
} catch (e) {
if (typeof FS == 'undefined' || !(e instanceof FS.ErrnoError)) throw e;
return -e(errno);
}
}

function __syscall_newfstatat(dirfd, path, buf, flags) {
try {

    path = SYSCALLS.getStr(path);
    var nofollow = flags & 256;
    var allowEmpty = flags & 4096;
    flags = flags & (~4352);
    assert(!flags, flags);
    path = SYSCALLS.calculateAt(dirfd, path, allowEmpty);
    return SYSCALLS.doStat(nofollow ? FS.lstat : FS.stat, path, buf);
} catch (e) {
if (typeof FS == 'undefined' || !(e instanceof FS.ErrnoError)) throw e;
return -e(errno);
}
}

function __syscall_openat(dirfd, path, flags, varargs) {
SYSCALLS.varargs = varargs;
try {

    path = SYSCALLS.getStr(path);
    path = SYSCALLS.calculateAt(dirfd, path);
    var mode = varargs ? SYSCALLS.get() : 0;
    return FS.open(path, flags, mode).fd;
} catch (e) {
if (typeof FS == 'undefined' || !(e instanceof FS.ErrnoError)) throw e;
return -e(errno);
}
}

function __syscall_readlinkat(dirfd, path, buf, bufsize) {

```

```

try {
    path = SYSCALLS.getStr(path);
    path = SYSCALLS.calculateAt(dirfd, path);
    return SYSCALLS.doReadlink(path, buf, bufsize);
} catch (e) {
    if (typeof FS == 'undefined' || !(e instanceof FS.ErrnoError)) throw e;
    return -e(errno);
}
}

function __syscall_recvfrom(fd, buf, len, flags, addr, addrlen) {
try {

    var sock = getSocketFromFD(fd);
    var msg = sock.sock_ops.recvmsg(sock, len);
    if (!msg) return 0; // socket is closed
    if (addr) {
        var errno = writeSockaddr(addr, sock.family, DNS.lookup_name(msg.addr),
msg.port, addrlen);
        assert(!errno);
    }
    HEAPU8.set(msg.buffer, buf);
    return msg.buffer.byteLength;
} catch (e) {
    if (typeof FS == 'undefined' || !(e instanceof FS.ErrnoError)) throw e;
    return -e(errno);
}
}

function __syscall_renameat(olddirfd, oldpath, newdirfd, newpath) {
try {

    oldpath = SYSCALLS.getStr(oldpath);
    newpath = SYSCALLS.getStr(newpath);
    oldpath = SYSCALLS.calculateAt(olddirfd, oldpath);
    newpath = SYSCALLS.calculateAt(newdirfd, newpath);
    FS.rename(oldpath, newpath);
    return 0;
} catch (e) {
    if (typeof FS == 'undefined' || !(e instanceof FS.ErrnoError)) throw e;
    return -e(errno);
}
}

function __syscall_rmdir(path) {
try {

    path = SYSCALLS.getStr(path);
    FS.rmdir(path);
    return 0;
} catch (e) {
    if (typeof FS == 'undefined' || !(e instanceof FS.ErrnoError)) throw e;
    return -e(errno);
}
}

```

```

}

function __syscall_sendto(fd, message, length, flags, addr, addr_len) {
try {

    var sock = getSocketFromFD(fd);
    var dest = getSocketAddress(addr, addr_len, true);
    if (!dest) {
        // send, no address provided
        return FS.write(sock.stream, HEAP8,message, length);
    } else {
        // sendto an address
        return sock.sock_ops.sendmsg(sock, HEAP8,message, length, dest.addr,
dest.port);
    }
} catch (e) {
if (typeof FS == 'undefined' || !(e instanceof FS.ErrnoError)) throw e;
return -e(errno);
}
}

function __syscall_socket(domain, type, protocol) {
try {

    var sock = SOCKFS.createSocket(domain, type, protocol);
    assert(sock.stream.fd < 64); // XXX ? select() assumes socket fd values are in
0..63
    return sock.stream.fd;
} catch (e) {
if (typeof FS == 'undefined' || !(e instanceof FS.ErrnoError)) throw e;
return -e(errno);
}
}

function __syscall_stat64(path, buf) {
try {

    path = SYSCALLS.getPath(path);
    return SYSCALLS.doStat(FS.stat, path, buf);
} catch (e) {
if (typeof FS == 'undefined' || !(e instanceof FS.ErrnoError)) throw e;
return -e(errno);
}
}

function __syscall_statfs64(path, size, buf) {
try {

    path = SYSCALLS.getPath(path);
    assert(size === 64);
    // NOTE: None of the constants here are true. We're just returning safe and
    //       sane values.
    HEAP32[((buf)+(4))>>2] = 4096;
    HEAP32[((buf)+(40))>>2] = 4096;
    HEAP32[((buf)+(8))>>2] = 1000000;
}
}
}
```

```

    HEAP32[((buf)+(12))>>2] = 500000;
    HEAP32[((buf)+(16))>>2] = 500000;
    HEAP32[((buf)+(20))>>2] = FS.nextInode;
    HEAP32[((buf)+(24))>>2] = 1000000;
    HEAP32[((buf)+(28))>>2] = 42;
    HEAP32[((buf)+(44))>>2] = 2; // ST_NOSUID
    HEAP32[((buf)+(36))>>2] = 255;
    return 0;
} catch (e) {
if (typeof FS == 'undefined' || !(e instanceof FS.ErrnoError)) throw e;
return -e(errno);
}
}

function __syscall_truncate64(path, low, high) {
try {

    path = SYSCALLS.getStr(path);
    var length = SYSCALLS.get64(low, high);
    FS.truncate(path, length);
    return 0;
} catch (e) {
if (typeof FS == 'undefined' || !(e instanceof FS.ErrnoError)) throw e;
return -e(errno);
}
}

function __syscall_unlinkat(dirfd, path, flags) {
try {

    path = SYSCALLS.getStr(path);
    path = SYSCALLS.calculateAt(dirfd, path);
    if (flags === 0) {
        FS.unlink(path);
    } else if (flags === 512) {
        FS.rmdir(path);
    } else {
        abort('Invalid flags passed to unlinkat');
    }
    return 0;
} catch (e) {
if (typeof FS == 'undefined' || !(e instanceof FS.ErrnoError)) throw e;
return -e(errno);
}
}

function __syscall_utimensat(dirfd, path, times, flags) {
try {

    path = SYSCALLS.getStr(path);
    assert(flags === 0);
    path = SYSCALLS.calculateAt(dirfd, path, true);
    if (!times) {
        var atime = Date.now();
        var mtime = atime;

```

```

    } else {
        var seconds = HEAP32[((times)>>2)];
        var nanoseconds = HEAP32[((((times)+(4))>>2))];
        atime = (seconds*1000) + (nanoseconds/(1000*1000));
        times += 8;
        seconds = HEAP32[((times)>>2)];
        nanoseconds = HEAP32[((((times)+(4))>>2))];
        mtime = (seconds*1000) + (nanoseconds/(1000*1000));
    }
    FS.utime(path, atime, mtime);
    return 0;
} catch (e) {
if (typeof FS == 'undefined' || !(e instanceof FS.ErrnoError)) throw e;
return -e(errno);
}
}

var dlopen_main_init = 0;
function __dlopen_js(handle) {
    warnOnce('Unable to open DLL! Dynamic linking is not supported in
WebAssembly builds due to limitations to performance and code size. Please statically
link in the needed libraries.');
    // Do not abort here - IL2CPP will throw a managed exception.

    // Return dummy success for the first dlopen since that is the __main__
module (so it gets past its assert checks),
    // and false otherwise. TODO: After Emscripten is updated to a version
newer than 3.1.8-unity, this logic can be
    // dropped: https://github.com/emscripten-core/emscripten/issues/16790
    var ret = !dlopen_main_init;
    dlopen_main_init = 1;
    return ret;
}

function __dlsym_js(handle, symbol) {
    return 0;
}

function __emscripten_date_now() {
    return Date.now();
}

var nowIsMonotonic = true;;
function __emscripten_get_now_is_monotonic() {
    return nowIsMonotonic;
}

function __emscripten_throw_longjmp() { throw Infinity; }

function __gmtime_js(time, tmPtr) {
    var date = new Date(HEAP32[((time)>>2)]*1000);
    HEAP32[((tmPtr)>>2)] = date.getUTCSeconds();
    HEAP32[((((tmPtr)+(4))>>2))] = date.getUTCMMinutes();
    HEAP32[((((tmPtr)+(8))>>2))] = date.getUTCHours();
    HEAP32[((((tmPtr)+(12))>>2))] = date.getUTCDate();
}

```

```

    HEAP32[((tmPtr)+(16))>>2] = date.getUTCMonth();
    HEAP32[((tmPtr)+(20))>>2] = date.getUTCFullYear()-1900;
    HEAP32[((tmPtr)+(24))>>2] = date.getUTCDay();
    var start = Date.UTC(date.getUTCFullYear(), 0, 1, 0, 0, 0, 0);
    var yday = ((date.getTime() - start) / (1000 * 60 * 60 * 24))|0;
    HEAP32[((tmPtr)+(28))>>2] = yday;
}

function __localtime_js(time, tmPtr) {
    var date = new Date(HEAP32[((time)>>2)]*1000);
    HEAP32[((tmPtr)>>2)] = date.getSeconds();
    HEAP32[((tmPtr)+(4))>>2] = date.getMinutes();
    HEAP32[((tmPtr)+(8))>>2] = date.getHours();
    HEAP32[((tmPtr)+(12))>>2] = date.getDate();
    HEAP32[((tmPtr)+(16))>>2] = date.getMonth();
    HEAP32[((tmPtr)+(20))>>2] = date.getFullYear()-1900;
    HEAP32[((tmPtr)+(24))>>2] = date.getDay();

    var start = new Date(date.getFullYear(), 0, 1);
    var yday = ((date.getTime() - start.getTime()) / (1000 * 60 * 60 * 24))|0;
    HEAP32[((tmPtr)+(28))>>2] = yday;
    HEAP32[((tmPtr)+(36))>>2] = -(date.getTimezoneOffset() * 60);

    // Attention: DST is in December in South, and some regions don't have DST at
    all.
    var summerOffset = new Date(date.getFullYear(), 6, 1).getTimezoneOffset();
    var winterOffset = start.getTimezoneOffset();
    var dst = (summerOffset != winterOffset && date.getTimezoneOffset() ==
    Math.min(winterOffset, summerOffset))|0;
    HEAP32[((tmPtr)+(32))>>2] = dst;
}

function __mktime_js(tmPtr) {
    var date = new Date(HEAP32[((tmPtr)+(20))>>2]) + 1900,
        HEAP32[((tmPtr)+(16))>>2],
        HEAP32[((tmPtr)+(12))>>2],
        HEAP32[((tmPtr)+(8))>>2],
        HEAP32[((tmPtr)+(4))>>2],
        HEAP32[((tmPtr)>>2)],
        0;

    // There's an ambiguous hour when the time goes back; the tm_isdst field is
    // used to disambiguate it. Date() basically guesses, so we fix it up if it
    // guessed wrong, or fill in tm_isdst with the guess if it's -1.
    var dst = HEAP32[((tmPtr)+(32))>>2];
    var guessedOffset = date.getTimezoneOffset();
    var start = new Date(date.getFullYear(), 0, 1);
    var summerOffset = new Date(date.getFullYear(), 6, 1).getTimezoneOffset();
    var winterOffset = start.getTimezoneOffset();
    var dstOffset = Math.min(winterOffset, summerOffset); // DST is in December in
South
    if (dst < 0) {
        // Attention: some regions don't have DST at all.
        HEAP32[((tmPtr)+(32))>>2] = Number(summerOffset != winterOffset && dstOffset
== guessedOffset);
    }
}

```

```

} else if ((dst > 0) != (dstOffset == guessedOffset)) {
    var nonDstOffset = Math.max(winterOffset, summerOffset);
    var trueOffset = dst > 0 ? dstOffset : nonDstOffset;
    // Don't try setMinutes(date.getMinutes() + ...) -- it's messed up.
    date.setTime(date.getTime() + (trueOffset - guessedOffset)*60000);
}

HEAP32[((tmPtr)+(24))>>2] = date.getDay();
var yday = ((date.getTime() - start.getTime()) / (1000 * 60 * 60 * 24))|0;
HEAP32[((tmPtr)+(28))>>2] = yday;
// To match expected behavior, update fields from date
HEAP32[((tmPtr)>>2)] = date.getSeconds();
HEAP32[((tmPtr)+(4))>>2] = date.getMinutes();
HEAP32[((tmPtr)+(8))>>2] = date.getHours();
HEAP32[((tmPtr)+(12))>>2] = date.getDate();
HEAP32[((tmPtr)+(16))>>2] = date.getMonth();

return (date.getTime() / 1000)|0;
}

function __mmap_js(addr, len, prot, flags, fd, off, allocated, builtin) {
try {

    var info = FS.getStream(fd);
    if (!info) return -8;
    var res = FS.mmap(info, addr, len, off, prot, flags);
    var ptr = res.ptr;
    HEAP32[(allocated)>>2] = res.allocated;
    return ptr;
} catch (e) {
    if (typeof FS == 'undefined' || !(e instanceof FS.ErrnoError)) throw e;
    return -e(errno);
}
}

function __munmap_js(addr, len, prot, flags, fd, offset) {
try {

    var stream = FS.getStream(fd);
    if (stream) {
        if (prot & 2) {
            SYSCALLS.doMsync(addr, stream, len, flags, offset);
        }
        FS.munmap(stream);
    }
} catch (e) {
    if (typeof FS == 'undefined' || !(e instanceof FS.ErrnoError)) throw e;
    return -e(errno);
}
}

function _tzset_impl(timezone, daylight, tzname) {
    var currentYear = new Date().getFullYear();
    var winter = new Date(currentYear, 0, 1);
    var summer = new Date(currentYear, 6, 1);
}

```

```

var winterOffset = winter.getTimezoneOffset();
var summerOffset = summer.getTimezoneOffset();

// Local standard timezone offset. Local standard time is not adjusted for
daylight savings.
// This code uses the fact that getTimezoneOffset returns a greater value during
Standard Time versus Daylight Saving Time (DST).
// Thus it determines the expected output during Standard Time, and it compares
whether the output of the given date the same (Standard) or less (DST).
var stdTimezoneOffset = Math.max(winterOffset, summerOffset);

// timezone is specified as seconds west of UTC ("The external variable
// `timezone` shall be set to the difference, in seconds, between
// Coordinated Universal Time (UTC) and local standard time."), the same
// as returned by stdTimezoneOffset.
// See http://pubs.opengroup.org/onlinepubs/009695399/functions/tzset.html
HEAP32[((timezone)>>2)] = stdTimezoneOffset * 60;

HEAP32[((daylight)>>2)] = Number(winterOffset != summerOffset);

function extractZone(date) {
    var match = date.toTimeString().match(/\(([A-Za-z ]+)\)\$/);
    return match ? match[1] : "GMT";
};

var winterName = extractZone(winter);
var summerName = extractZone(summer);
var winterNamePtr = allocateUTF8(winterName);
var summerNamePtr = allocateUTF8(summerName);
if (summerOffset < winterOffset) {
    // Northern hemisphere
    HEAP32[((tzname)>>2)] = winterNamePtr;
    HEAP32[((tzname)+(4))>>2] = summerNamePtr;
} else {
    HEAP32[((tzname)>>2)] = summerNamePtr;
    HEAP32[((tzname)+(4))>>2] = winterNamePtr;
}
}

function __tzset_js(timezone, daylight, tzname) {
    // TODO: Use (malleable) environment variables instead of system settings.
    if (__tzset_js.called) return;
    __tzset_js.called = true;
    __tzset_impl(timezone, daylight, tzname);
}

function __abort() {
    abort('native code called abort()');
}

var readAsmConstArgsArray = [];
function readAsmConstArgs(sigPtr, buf) {
    ;
    // Nobody should have mutated __readAsmConstArgsArray underneath us to be
something else than an array.
    assert(Array.isArray(readAsmConstArgsArray));
    // The input buffer is allocated on the stack, so it must be stack-aligned.
}

```

```

assert(buf % 16 == 0);
readAsmConstArgsArray.length = 0;
var ch;
// Most arguments are i32s, so shift the buffer pointer so it is a plain
// index into HEAP32.
buf >>= 2;
while (ch = HEAPU8[sigPtr++]) {
    assert(ch === 100/*'d'*/ || ch === 102/*'f'*/ || ch === 105 /*'i'*/, 'Invalid
character ' + ch + '(' + String.fromCharCode(ch) + ')' in readAsmConstArgs! Use only
"d", "f" or "i", and do not specify "v" for void return argument.');
    // A double takes two 32-bit slots, and must also be aligned - the backend
    // will emit padding to avoid that.
    var readAsmConstArgsDouble = ch < 105;
    if (readAsmConstArgsDouble && (buf & 1)) buf++;
    readAsmConstArgsArray.push(readAsmConstArgsDouble ? HEAPF64[buf++ >> 1] :
HEAP32[buf]);
    ++buf;
}
return readAsmConstArgsArray;
}

function _emscripten_asm_const_int(code, sigPtr, argbuf) {
    var args = readAsmConstArgs(sigPtr, argbuf);
    if (!ASM_CONSTS.hasOwnProperty(code)) abort('No EM_ASM constant found at address
' + code);
    return ASM_CONSTS[code].apply(null, args);
}

function mainThreadEM_ASM(code, sigPtr, argbuf, sync) {
    var args = readAsmConstArgs(sigPtr, argbuf);
    if (!ASM_CONSTS.hasOwnProperty(code)) abort('No EM_ASM constant found at address
' + code);
    return ASM_CONSTS[code].apply(null, args);
}

function _emscripten_asm_const_int_sync_on_main_thread(code, sigPtr, argbuf) {
    return mainThreadEM_ASM(code, sigPtr, argbuf, 1);
}

function _emscripten_set_main_loop_timing(mode, value) {
    Browser.mainLoop.timingMode = mode;
    Browser.mainLoop.timingValue = value;

    if (!Browser.mainLoop.func) {
        err('emscripten_set_main_loop_timing: Cannot set timing mode for main loop
since a main loop does not exist! Call emscripten_set_main_loop first to set one up.');
        return 1; // Return non-zero on failure, can't set timing mode when there is no
main loop.
    }

    if (!Browser.mainLoop.running) {

        Browser.mainLoop.running = true;
    }
    if (mode == 0 /*EM_TIMING_SETTIMEOUT*/) {
        Browser.mainLoop.scheduler = function Browser_mainLoop_scheduler setTimeout() {

```

```

        var timeUntilNextTick = Math.max(0, Browser.mainLoop.tickStartTime + value -
_emscripten_get_now())|0;
        setTimeout(Browser.mainLoop.runner, timeUntilNextTick); // doing this each
time means that on exception, we stop
    };
    Browser.mainLoop.method = 'timeout';
} else if (mode == 1 /*EM_TIMING_RAF*/) {
    Browser.mainLoop.scheduler = function Browser_mainLoop_scheduler_rAF() {
        Browser.requestAnimationFrame(Browser.mainLoop.runner);
    };
    Browser.mainLoop.method = 'rAF';
} else if (mode == 2 /*EM_TIMING_SETIMMEDIATE*/) {
    if (typeof setImmediate == 'undefined') {
        // Emulate setImmediate. (note: not a complete polyfill, we don't emulate
clearImmediate() to keep code size to minimum, since not needed)
        var setImmediates = [];
        var emscriptenMainLoopMessageId = 'setimmediate';
        var Browser_setImmediate_messageHandler = function/** @type {Event} */
event) {
            // When called in current thread or Worker, the main loop ID is structured
slightly different to accommodate for --proxy-to-worker runtime listening to Worker
events,
            // so check for both cases.
            if (event.data === emscriptenMainLoopMessageId || event.data.target ===
emscriptenMainLoopMessageId) {
                event.stopPropagation();
                setImmediates.shift();
            }
        }
        addEventListener("message", Browser_setImmediate_messageHandler, true);
        setImmediate = /** @type{function(function(): ?, ...?: number)} */(function
Browser_emulated_setImmediate(func) {
            setImmediates.push(func);
            if (ENVIRONMENT_IS_WORKER) {
                if (Module['setImmediates'] === undefined) Module['setImmediates'] = [];
                Module['setImmediates'].push(func);
                postMessage({target: emscriptenMainLoopMessageId}); // In --proxy-to-
worker, route the message via proxyClient.js
            } else postMessage(emscriptenMainLoopMessageId, "*"); // On the main
thread, can just send the message to itself.
            })
        }
        Browser.mainLoop.scheduler = function Browser_mainLoop_scheduler_setImmediate()
{
            setImmediate(Browser.mainLoop.runner);
        };
        Browser.mainLoop.method = 'immediate';
    }
    return 0;
}

var _emscripten_get_now;if (ENVIRONMENT_IS_NODE) {
    _emscripten_get_now = () => {
        var t = process['hrtime']();
        return t[0] * 1e3 + t[1] / 1e6;
    };
}

```

```

} else _emscripten_get_now = () => performance.now();
;

function runtimeKeepalivePush() {
}

function _exit(status) {
    // void exit(int status);
    // http://pubs.opengroup.org/onlinepubs/000095399/functions/exit.html
    exit(status);
}
function maybeExit() {
}

/**
 * @param {number=} arg
 * @param {boolean=} noSetTiming
 */
function setMainLoop(browserIterationFunc, fps, simulateInfiniteLoop, arg,
noSetTiming) {
    assert(!Browser.mainLoop.func, 'emscripten_set_main_loop: there can only be one
main loop function at once: call emscripten_cancel_main_loop to cancel the previous one
before setting a new one with different parameters.');

    Browser.mainLoop.func = browserIterationFunc;
    Browser.mainLoop.arg = arg;

    var thisMainLoopId = Browser.mainLoop.currentlyRunningMainloop;
    function checkIsRunning() {
        if (thisMainLoopId < Browser.mainLoop.currentlyRunningMainloop) {

            maybeExit();
            return false;
        }
        return true;
    }

    // We create the loop runner here but it is not actually running until
    // _emscripten_set_main_loop_timing is called (which might happen a
    // later time). This member signifies that the current runner has not
    // yet been started so that we can call runtimeKeepalivePush when it
    // gets its timing set for the first time.
    Browser.mainLoop.running = false;
    Browser.mainLoop.runner = function Browser_mainLoop_runner() {
        if (ABORT) return;
        if (Browser.mainLoop.queue.length > 0) {
            var start = Date.now();
            var blocker = Browser.mainLoop.queue.shift();
            blocker.func(blocker.arg);
            if (Browser.mainLoop.remainingBlockers) {
                var remaining = Browser.mainLoop.remainingBlockers;
                var next = remaining%1 == 0 ? remaining-1 : Math.floor(remaining);
                if (blocker.counted) {
                    Browser.mainLoop.remainingBlockers = next;
                } else {

```

```

        // not counted, but move the progress along a tiny bit
        next = next + 0.5; // do not steal all the next one's progress
        Browser.mainLoop.remainingBlockers = (8*remaining + next)/9;
    }
}

out('main loop blocker "' + blocker.name + '" took ' + (Date.now() - start) +
' ms'); //, left: ' + Browser.mainLoop.remainingBlockers);
Browser.mainLoop.updateStatus();

// catches pause/resume main loop from blocker execution
if (!checkIsRunning()) return;

setTimeout(Browser.mainLoop.runner, 0);
return;
}

// catch pauses from non-main loop sources
if (!checkIsRunning()) return;

// Implement very basic swap interval control
Browser.mainLoop.currentFrameNumber = Browser.mainLoop.currentFrameNumber + 1 |
0;
if (Browser.mainLoop.timingMode == 1/*EM_TIMING_RAF*/ &&
Browser.mainLoop.timingValue > 1 && Browser.mainLoop.currentFrameNumber %
Browser.mainLoop.timingValue != 0) {
    // Not the scheduled time to render this frame - skip.
    Browser.mainLoop.scheduler();
    return;
} else if (Browser.mainLoop.timingMode == 0/*EM_TIMING_SETTIMEOUT*/) {
    Browser.mainLoop.tickStartTime = _emscripten_get_now();
}

// Signal GL rendering layer that processing of a new frame is about to start.
// This helps it optimize
// VBO double-buffering and reduce GPU stalls.
GL.newRenderingFrameStarted();

if (Browser.mainLoop.method === 'timeout' && Module.ctx) {
    warnOnce('Looks like you are rendering without using requestAnimationFrame
for the main loop. You should use 0 for the frame rate in emscripten_set_main_loop
in order to use requestAnimationFrame, as that can greatly improve your frame rates!');
    Browser.mainLoop.method = ''; // just warn once per call to set main loop
}

Browser.mainLoop.runIter(browserIterationFunc);

checkStackCookie();

// catch pauses from the main loop itself
if (!checkIsRunning()) return;

// Queue new audio data. This is important to be right after the main loop
invocation, so that we will immediately be able
// to queue the newest produced audio samples.

```

```

    // TODO: Consider adding pre- and post- rAF callbacks so that
    GL.newRenderingFrameStarted() and SDL.audio.queueNewAudioData()
        //           do not need to be hardcoded into this function, but can be more
generic.
    if (typeof SDL == 'object' && SDL.audio && SDL.audio.queueNewAudioData)
SDL.audio.queueNewAudioData();

    Browser.mainLoop.scheduler();
}

if (!noSetTiming) {
    if (fps && fps > 0) _emscripten_set_main_loop_timing(0/*EM_TIMING_SETTIMEOUT*/,
1000.0 / fps);
    else _emscripten_set_main_loop_timing(1/*EM_TIMING_RAF*/, 1); // Do rAF by
rendering each frame (no decimating)

    Browser.mainLoop.scheduler();
}

if (simulateInfiniteLoop) {
    throw 'unwind';
}
}

/** @param {boolean=} synchronous */
function callUserCallback(func, synchronous) {
    if (ABORT) {
        err('user callback triggered after runtime exited or application aborted.
Ignoring.');
        return;
    }
    // For synchronous calls, let any exceptions propagate, and don't let the runtime
exit.
    if (synchronous) {
        func();
        return;
    }
    try {
        func();
    } catch (e) {
        handleException(e);
    }
}

function runtimeKeepalivePop() {
}

/** @param {number=} timeout */
function safesetTimeout(func, timeout) {

    return setTimeout(function() {
        callUserCallback(func);
    }, timeout);
}

```

```

var Browser =
{mainLoop:{running:false,scheduler:null,method:"",currentlyRunningMainloop:0,func:null,
arg:0,timingMode:0,timingValue:0,currentFrameNumber:0,queue:[],pause:function() {
    Browser.mainLoop.scheduler = null;
    // Incrementing this signals the previous main loop that it's now become old,
and it must return.
    Browser.mainLoop.currentlyRunningMainloop++;
},resume:function() {
    Browser.mainLoop.currentlyRunningMainloop++;
    var timingMode = Browser.mainLoop.timingMode;
    var timingValue = Browser.mainLoop.timingValue;
    var func = Browser.mainLoop.func;
    Browser.mainLoop.func = null;
    // do not set timing and call scheduler, we will do it on the next lines
    setMainLoop(func, 0, false, Browser.mainLoop.arg, true);
    _emscripten_set_main_loop_timing(timingMode, timingValue);
    Browser.mainLoop.scheduler();
},updateStatus:function() {
    if (Module['setStatus']) {
        var message = Module['statusMessage'] || 'Please wait...';
        var remaining = Browser.mainLoop.remainingBlockers;
        var expected = Browser.mainLoop.expectedBlockers;
        if (remaining) {
            if (remaining < expected) {
                Module['setStatus'](message + ' (' + (expected - remaining) + '/' +
expected + ')');
            } else {
                Module['setStatus'](message);
            }
        } else {
            Module['setStatus']('');
        }
    }
},runIter:function(func) {
    if (ABORT) return;
    if (Module['preMainLoop']) {
        var preRet = Module['preMainLoop']();
        if (preRet === false) {
            return; // |return false| skips a frame
        }
    }
    callUserCallback(func);
    if (Module['postMainLoop']) Module['postMainLoop']();
}},isFullscreen:false,pointerLock:false,moduleContextCreatedCallbacks:[],workers:[],ini
t:function() {
    if (!Module["preloadPlugins"]) Module["preloadPlugins"] = []; // needs to exist
even in workers

    if (Browser.initiated) return;
    Browser.initiated = true;

    try {
        new Blob();
        Browser.hasBlobConstructor = true;
    } catch(e) {
        Browser.hasBlobConstructor = false;
        out("warning: no blob constructor, cannot create blobs with mimetypes");
    }
}

```

```

        }

        Browser.BlobBuilder = typeof MozBlobBuilder != "undefined" ? MozBlobBuilder :
(typeof WebKitBlobBuilder != "undefined" ? WebKitBlobBuilder :
(!Browser.hasBlobConstructor ? out("warning: no BlobBuilder") : null));
        Browser.URLObject = typeof window != "undefined" ? (window.URL ? window.URL :
window.webkitURL) : undefined;
        if (!Module.noImageDecoding && typeof Browser.URLObject == 'undefined') {
            out("warning: Browser does not support creating object URLs. Built-in browser
image decoding will not be available.");
            Module.noImageDecoding = true;
        }

        // Support for plugins that can process preloaded files. You can add more of
these to
        // your app by creating and appending to Module.preloadPlugins.
        //
        // Each plugin is asked if it can handle a file based on the file's name. If it
can,
        // it is given the file's raw data. When it is done, it calls a callback with
the file's
        // (possibly modified) data. For example, a plugin might decompress a file, or
it
        // might create some side data structure for use later (like an Image element,
etc.).

        var imagePlugin = {};
        imagePlugin['canHandle'] = function imagePlugin_canHandle(name) {
            return !Module.noImageDecoding && /\.(jpg|jpeg|png|bmp)$/.i.test(name);
        };
        imagePlugin['handle'] = function imagePlugin_handle(byteArray, name, onload,
onerror) {
            var b = null;
            if (Browser.hasBlobConstructor) {
                try {
                    b = new Blob([byteArray], { type: Browser.getMimetype(name) });
                    if (b.size !== byteArray.length) { // Safari bug #118630
                        // Safari's Blob can only take an ArrayBuffer
                        b = new Blob([(new Uint8Array(byteArray)).buffer], { type:
Browser.getMimetype(name) });
                    }
                } catch(e) {
                    warnOnce('Blob constructor present but fails: ' + e + '; falling back to
blob builder');
                }
            }
            if (!b) {
                var bb = new Browser.BlobBuilder();
                bb.append((new Uint8Array(byteArray)).buffer); // we need to pass a buffer,
and must copy the array to get the right data range
                b = bb.getBlob();
            }
            var url = Browser.URLObject.createObjectURL(b);
            assert(typeof url == 'string', 'createObjectURL must return a url as a
string');
            var img = new Image();

```

```

    img.onload = () => {
        assert(img.complete, 'Image ' + name + ' could not be decoded');
        var canvas = /** @type {!HTMLCanvasElement} */
(document.createElement('canvas'));
        canvas.width = img.width;
        canvas.height = img.height;
        var ctx = canvas.getContext('2d');
        ctx.drawImage(img, 0, 0);
        Module["preloadedImages"][name] = canvas;
        Browser.URLObject.revokeObjectURL(url);
        if (onload) onload(byteArray);
    };
    img.onerror = (event) => {
        out('Image ' + url + ' could not be decoded');
        if (onerror) onerror();
    };
    img.src = url;
};

Module['preloadPlugins'].push(imagePlugin);

var audioPlugin = {};
audioPlugin['canHandle'] = function audioPlugin_canHandle(name) {
    return !Module.noAudioDecoding && name.substr(-4) in { '.ogg': 1, '.wav': 1,
'.mp3': 1 };
};
audioPlugin['handle'] = function audioPlugin_handle(byteArray, name, onload,
onerror) {
    var done = false;
    function finish(audio) {
        if (done) return;
        done = true;
        Module["preloadedAudios"][name] = audio;
        if (onload) onload(byteArray);
    }
    function fail() {
        if (done) return;
        done = true;
        Module["preloadedAudios"][name] = new Audio(); // empty shim
        if (onerror) onerror();
    }
    if (Browser.hasBlobConstructor) {
        try {
            var b = new Blob([byteArray], { type: Browser.getMimetype(name) });
        } catch(e) {
            return fail();
        }
        var url = Browser.URLObject.createObjectURL(b); // XXX we never revoke
this!
        assert(typeof url == 'string', 'createObjectURL must return a url as a
string');
        var audio = new Audio();
        audio.addEventListener('canplaythrough', function() { finish(audio) },
false); // use addEventListener due to chromium bug 124926
        audio.onerror = function audio_onerror(event) {
            if (done) return;

```

```

        out('warning: browser could not fully decode audio ' + name + ', trying
slower base64 approach');
        function encode64(data) {
            var BASE =
'ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz0123456789+/';
            var PAD = '=';
            var ret = '';
            var leftchar = 0;
            var leftbits = 0;
            for (var i = 0; i < data.length; i++) {
                leftchar = (leftchar << 8) | data[i];
                leftbits += 8;
                while (leftbits >= 6) {
                    var curr = (leftchar >> (leftbits-6)) & 0x3f;
                    leftbits -= 6;
                    ret += BASE[curr];
                }
            }
            if (leftbits == 2) {
                ret += BASE[(leftchar&3) << 4];
                ret += PAD + PAD;
            } else if (leftbits == 4) {
                ret += BASE[(leftchar&0xf) << 2];
                ret += PAD;
            }
            return ret;
        }
        audio.src = 'data:audio/x-' + name.substr(-3) + ';base64,' +
encode64(byteArray);
        finish(audio); // we don't wait for confirmation this worked - but it's
worth trying
    };
    audio.src = url;
    // workaround for chrome bug 124926 - we do not always get oncanplaythrough
or onerror
    safeSetTimeout(function() {
        finish(audio); // try to use it even though it is not necessarily ready
to play
    }, 10000);
} else {
    return fail();
}
};

Module['preloadPlugins'].push(audioPlugin);

// Canvas event setup

function pointerLockChange() {
    Browser.pointerLock = document['pointerLockElement'] === Module['canvas'] ||
                        document['mozPointerLockElement'] === Module['canvas']
    ||
    document['webkitPointerLockElement'] ===
    Module['canvas'] ||
    document['msPointerLockElement'] === Module['canvas'];
}

```

```

var canvas = Module['canvas'];
if (canvas) {
    // forced aspect ratio can be enabled by defining 'forcedAspectRatio' on
Module
    // Module['forcedAspectRatio'] = 4 / 3;

    canvas.requestPointerLock = canvas['requestPointerLock'] ||
                                canvas['mozRequestPointerLock'] ||
                                canvas['webkitRequestPointerLock'] ||
                                canvas['msRequestPointerLock'] ||
                                function() {};
    canvas.exitPointerLock = document['exitPointerLock'] ||
                            document['mozExitPointerLock'] ||
                            document['webkitExitPointerLock'] ||
                            document['msExitPointerLock'] ||
                            function(){}; // no-op if function does not exist
    canvas.exitPointerLock = canvas.exitPointerLock.bind(document);

    document.addEventListener('pointerlockchange', pointerLockChange, false);
    document.addEventListener('mozpointerlockchange', pointerLockChange, false);
    document.addEventListener('webkitpointerlockchange', pointerLockChange,
false);
    document.addEventListener('mspointerlockchange', pointerLockChange, false);

    if (Module['elementPointerLock']) {
        canvas.addEventListener("click", function(ev) {
            if (!Browser.pointerLock && Module['canvas'].requestPointerLock) {
                Module['canvas'].requestPointerLock();
                ev.preventDefault();
            }
        }, false);
    }
},handledByPreloadPlugin:function(byteArray, fullname, finish, onerror) {
    // Ensure plugins are ready.
    Browser.init();

    var handled = false;
    Module['preloadPlugins'].forEach(function(plugin) {
        if (handled) return;
        if (plugin['canHandle'](fullname)) {
            plugin['handle'](byteArray, fullname, finish, onerror);
            handled = true;
        }
    });
    return handled;
},createContext:function(** @type {HTMLCanvasElement} */ canvas, useWebGL,
setInModule, webGLContextAttributes) {
    if (useWebGL && Module.ctx && canvas == Module.canvas) return Module.ctx; // no
need to recreate GL context if it's already been created for this canvas.

    var ctx;
    var contextHandle;
    if (useWebGL) {

```

```

    // For GLES2/desktop GL compatibility, adjust a few defaults to be different
    // to WebGL defaults, so that they align better with the desktop defaults.
    var contextAttributes = {
        antialias: false,
        alpha: false,
        majorVersion: (typeof WebGL2RenderingContext != 'undefined') ? 2 : 1,
    };

    if (webGLContextAttributes) {
        for (var attribute in webGLContextAttributes) {
            contextAttributes[attribute] = webGLContextAttributes[attribute];
        }
    }

    // This check of existence of GL is here to satisfy Closure compiler, which
    // yells if variable GL is referenced below but GL object is not
    // actually compiled in because application is not doing any GL operations.
    TODO: Ideally if GL is not being used, this function
        // Browser.createContext() should not even be emitted.
        if (typeof GL != 'undefined') {
            contextHandle = GL.createContext(canvas, contextAttributes);
            if (contextHandle) {
                ctx = GL.getContext(contextHandle).GLctx;
            }
        }
    } else {
        ctx = canvas.getContext('2d');
    }

    if (!ctx) return null;

    if (setInModule) {
        if (!useWebGL) assert(typeof GLctx == 'undefined', 'cannot set in module if
GLctx is used, but we are a non-GL context that would replace it');

        Module.ctx = ctx;
        if (useWebGL) GL.makeContextCurrent(contextHandle);
        Module.useWebGL = useWebGL;
        Browser.moduleContextCreatedCallbacks.forEach(function(callback) { callback()
    });
        Browser.init();
    }
    return ctx;
},destroyContext:function(canvas, useWebGL, setInModule)
{},fullscreenHandlersInstalled:false,lockPointer:undefined,resizeCanvas:undefined,reqe
stFullscreen:function(lockPointer, resizeCanvas) {
    Browser.lockPointer = lockPointer;
    Browser.resizeCanvas = resizeCanvas;
    if (typeof Browser.lockPointer == 'undefined') Browser.lockPointer = true;
    if (typeof Browser.resizeCanvas == 'undefined') Browser.resizeCanvas = false;

    var canvas = Module['canvas'];
    function fullscreenChange() {
        Browser.isFullscreen = false;
        var canvasContainer = canvas.parentNode;
        if ((document['fullscreenElement'] || document['mozFullScreenElement']) ||

```

```

        document['msFullscreenElement'] || document['webkitFullscreenElement']
    ||
        document['webkitCurrentFullScreenElement']) === canvasContainer) {
    canvas.exitFullscreen = Browser.exitFullscreen;
    if (Browser.lockPointer) canvas.requestPointerLock();
    Browser.isFullscreen = true;
    if (Browser.resizeCanvas) {
        Browser.setFullscreenCanvasSize();
    } else {
        Browser.updateCanvasDimensions(canvas);
    }
} else {
    // remove the full screen specific parent of the canvas again to restore
    the HTML structure from before going full screen
    canvasContainer.parentNode.insertBefore(canvas, canvasContainer);
    canvasContainer.parentNode.removeChild(canvasContainer);

    if (Browser.resizeCanvas) {
        Browser.setWindowedCanvasSize();
    } else {
        Browser.updateCanvasDimensions(canvas);
    }
}
if (Module['onFullScreen']) Module['onFullScreen'](Browser.isFullscreen);
if (Module['onFullscreen']) Module['onFullscreen'](Browser.isFullscreen);
}

if (!Browser.fullscreenHandlersInstalled) {
    Browser.fullscreenHandlersInstalled = true;
    document.addEventListener('fullscreenchange', fullscreenChange, false);
    document.addEventListener('mozfullscreenchange', fullscreenChange, false);
    document.addEventListener('webkitfullscreenchange', fullscreenChange, false);
    document.addEventListener('MSFullscreenChange', fullscreenChange, false);
}

// create a new parent to ensure the canvas has no siblings. this allows
// browsers to optimize full screen performance when its parent is the full screen root
var canvasContainer = document.createElement("div");
canvas.parentNode.insertBefore(canvasContainer, canvas);
canvasContainer.appendChild(canvas);

// use parent of canvas as full screen root to allow aspect ratio correction
// (Firefox stretches the root to screen size)
canvasContainer.requestFullscreen = canvasContainer['requestFullscreen'] ||
                                    canvasContainer['mozRequestFullScreen'] ||
                                    canvasContainer['msRequestFullscreen'] ||
                                    (canvasContainer['webkitRequestFullscreen'] ?
                                    function() {
                                        canvasContainer['webkitRequestFullscreen'](Element['ALLOW_KEYBOARD_INPUT']) : null)
                                    :
                                    (canvasContainer['webkitRequestFullScreen'] ?
                                    function() {
                                        canvasContainer['webkitRequestFullScreen'](Element['ALLOW_KEYBOARD_INPUT']) : null);

                                    canvasContainer.requestFullscreen();
},requestFullScreen:function() {

```

```

abort('Module.requestFullScreen has been replaced by Module.requestFullscreen
(without a capital S)');
},exitFullscreen:function() {
    // This is workaround for chrome. Trying to exit from fullscreen
    // not in fullscreen state will cause "TypeError: Document not active"
    // in chrome. See https://github.com/emscripten-core/emscripten/pull/8236
    if (!Browser.isFullscreen) {
        return false;
    }

    var CFS = document['exitFullscreen'] ||
              document['cancelFullScreen'] ||
              document['mozCancelFullScreen'] ||
              document['msExitFullscreen'] ||
              document['webkitCancelFullScreen'] ||
              (function() {});
    CFS.apply(document, []);
    return true;
},nextRAF:0,fakeRequestAnimationFrame:function(func) {
    // try to keep 60fps between calls to here
    var now = Date.now();
    if (Browser.nextRAF === 0) {
        Browser.nextRAF = now + 1000/60;
    } else {
        while (now + 2 >= Browser.nextRAF) { // fudge a little, to avoid timer jitter
causing us to do lots of delay:0
            Browser.nextRAF += 1000/60;
        }
    }
    var delay = Math.max(Browser.nextRAF - now, 0);
    setTimeout(func, delay);
},requestAnimationFrame:function(func) {
    if (typeof requestAnimationFrame == 'function') {
        requestAnimationFrame(func);
        return;
    }
    var RAF = Browser.fakeRequestAnimationFrame;
    RAF(func);
},safesetTimeout:function(func) {
    // Legacy function, this is used by the SDL2 port so we need to keep it
    // around at least until that is updated.
    return safesetTimeout(func);
},safeRequestAnimationFrame:function(func) {

    return Browser.requestAnimationFrame(function() {

        callUserCallback(func);
    });
},getMimetype:function(name) {
    return {
        'jpg': 'image/jpeg',
        'jpeg': 'image/jpeg',
        'png': 'image/png',
        'bmp': 'image/bmp',
        'ogg': 'audio/ogg',

```

```

'wav': 'audio/wav',
'mp3': 'audio/mpeg'
}[name.substr(name.lastIndexOf('.')+1)];
},getUserMedia:function(func) {
  if (!window.getUserMedia) {
    window.getUserMedia = navigator['getUserMedia'] ||
                          navigator['moz GetUserMedia'];
  }
  window.getUserMedia(func);
},getMovementX:function(event) {
  return event['movementX'] ||
         event['mozMovementX'] ||
         event['webkitMovementX'] ||
         0;
},getMovementY:function(event) {
  return event['movementY'] ||
         event['mozMovementY'] ||
         event['webkitMovementY'] ||
         0;
},getMouseWheelDelta:function(event) {
  var delta = 0;
  switch (event.type) {
    case 'DOMMouseScroll':
      // 3 lines make up a step
      delta = event.detail / 3;
      break;
    case 'mousewheel':
      // 120 units make up a step
      delta = event.wheelDelta / 120;
      break;
    case 'wheel':
      delta = event.deltaY
      switch (event.deltaMode) {
        case 0:
          // DOM_DELTA_PIXEL: 100 pixels make up a step
          delta /= 100;
          break;
        case 1:
          // DOM_DELTA_LINE: 3 lines make up a step
          delta /= 3;
          break;
        case 2:
          // DOM_DELTA_PAGE: A page makes up 80 steps
          delta *= 80;
          break;
        default:
          throw 'unrecognized mouse wheel delta mode: ' + event.deltaMode;
      }
      break;
    default:
      throw 'unrecognized mouse wheel event: ' + event.type;
  }
  return delta;
}

```

```

}, mouseX:0, mouseY:0, mouseMovementX:0, mouseMovementY:0, touches:{}, lastTouches:{}, calculateMouseEvent:function(event) { // event should be mousemove, mousedown or mouseup
    if (Browser.pointerLock) {
        // When the pointer is locked, calculate the coordinates
        // based on the movement of the mouse.
        // Workaround for Firefox bug 764498
        if (event.type != 'mousemove' &&
            ('mozMovementX' in event)) {
            Browser.mouseMovementX = Browser.mouseMovementY = 0;
        } else {
            Browser.mouseMovementX = Browser.getMovementX(event);
            Browser.mouseMovementY = Browser.getMovementY(event);
        }
    }

    // check if SDL is available
    if (typeof SDL != "undefined") {
        Browser.mouseX = SDL.mouseX + Browser.mouseMovementX;
        Browser.mouseY = SDL.mouseY + Browser.mouseMovementY;
    } else {
        // just add the mouse delta to the current absolute mouse position
        // FIXME: ideally this should be clamped against the canvas size and zero
        Browser.mouseX += Browser.mouseMovementX;
        Browser.mouseY += Browser.mouseMovementY;
    }
} else {
    // Otherwise, calculate the movement based on the changes
    // in the coordinates.
    var rect = Module["canvas"].getBoundingClientRect();
    var cw = Module["canvas"].width;
    var ch = Module["canvas"].height;

    // Neither .scrollX or .pageXOffset are defined in a spec, but
    // we prefer .scrollX because it is currently in a spec draft.
    // (see: http://www.w3.org/TR/2013/WD-cssom-view-20131217/)
    var scrollX = ((typeof window.scrollX != 'undefined') ? window.scrollX :
window.pageXOffset);
    var scrollY = ((typeof window.scrollY != 'undefined') ? window.scrollY :
window.pageYOffset);
    // If this assert lands, it's likely because the browser doesn't support
    scrollX or pageXOffset
    // and we have no viable fallback.
    assert((typeof scrollX != 'undefined') && (typeof scrollY != 'undefined'),
'Unable to retrieve scroll position, mouse positions likely broken.');

    if (event.type === 'touchstart' || event.type === 'touchend' || event.type
=== 'touchmove') {
        var touch = event.touch;
        if (touch === undefined) {
            return; // the "touch" property is only defined in SDL
        }
        var adjustedX = touch.pageX - (scrollX + rect.left);
        var adjustedY = touch.pageY - (scrollY + rect.top);

        adjustedX = adjustedX * (cw / rect.width);
        adjustedY = adjustedY * (ch / rect.height);
    }
}

```

```

var coords = { x: adjustedX, y: adjustedY };

if (event.type === 'touchstart') {
    Browser.lastTouches[touch.identifier] = coords;
    Browser.touches[touch.identifier] = coords;
} else if (event.type === 'touchend' || event.type === 'touchmove') {
    var last = Browser.touches[touch.identifier];
    if (!last) last = coords;
    Browser.lastTouches[touch.identifier] = last;
    Browser.touches[touch.identifier] = coords;
}
return;
}

var x = event.pageX - (scrollX + rect.left);
var y = event.pageY - (scrollY + rect.top);

// the canvas might be CSS-scaled compared to its backbuffer;
// SDL-using content will want mouse coordinates in terms
// of backbuffer units.
x = x * (cw / rect.width);
y = y * (ch / rect.height);

Browser.mouseMovementX = x - Browser.mousePosition;
Browser.mouseMovementY = y - Browser.mousePosition;
Browser.mousePosition = x;
Browser.mousePosition = y;
}
},resizeListeners:[],updateResizeListeners:function() {
var canvas = Module['canvas'];
Browser.resizeListeners.forEach(function(listener) {
    listener(canvas.width, canvas.height);
});
},setCanvasSize:function(width, height, noUpdates) {
var canvas = Module['canvas'];
Browser.updateCanvasDimensions(canvas, width, height);
if (!noUpdates) Browser.updateResizeListeners();
},windowedWidth:0>windowedHeight:0, setFullscreenCanvasSize:function() {
// check if SDL is available
if (typeof SDL != "undefined") {
    var flags = HEAPU32[((SDL.screen)>>2)];
    flags = flags | 0x00800000; // set SDL_FULLSCREEN flag
    HEAP32[((SDL.screen)>>2)] = flags;
}
Browser.updateCanvasDimensions(Module['canvas']);
Browser.updateResizeListeners();
},setWindowedCanvasSize:function() {
// check if SDL is available
if (typeof SDL != "undefined") {
    var flags = HEAPU32[((SDL.screen)>>2)];
    flags = flags & ~0x00800000; // clear SDL_FULLSCREEN flag
    HEAP32[((SDL.screen)>>2)] = flags;
}
Browser.updateCanvasDimensions(Module['canvas']);
}

```

```

        Browser.updateResizeListeners();
    }, updateCanvasDimensions:function(canvas, wNative, hNative) {
        if (wNative && hNative) {
            canvas.widthNative = wNative;
            canvas.heightNative = hNative;
        } else {
            wNative = canvas.widthNative;
            hNative = canvas.heightNative;
        }
        var w = wNative;
        var h = hNative;
        if (Module['forcedAspectRatio'] && Module['forcedAspectRatio'] > 0) {
            if (w/h < Module['forcedAspectRatio']) {
                w = Math.round(h * Module['forcedAspectRatio']);
            } else {
                h = Math.round(w / Module['forcedAspectRatio']);
            }
        }
        if (((document['fullscreenElement'] || document['mozFullScreenElement'] ||
              document['msFullscreenElement'] || document['webkitFullscreenElement']) ||
              document['webkitCurrentFullScreenElement']) === canvas.parentNode) &&
        (typeof screen != 'undefined')) {
            var factor = Math.min(screen.width / w, screen.height / h);
            w = Math.round(w * factor);
            h = Math.round(h * factor);
        }
        if (Browser.resizeCanvas) {
            if (canvas.width != w) canvas.width = w;
            if (canvas.height != h) canvas.height = h;
            if (typeof canvas.style != 'undefined') {
                canvas.style.removeProperty( "width");
                canvas.style.removeProperty("height");
            }
        } else {
            if (canvas.width != wNative) canvas.width = wNative;
            if (canvas.height != hNative) canvas.height = hNative;
            if (typeof canvas.style != 'undefined') {
                if (w != wNative || h != hNative) {
                    canvas.style.setProperty( "width", w + "px", "important");
                    canvas.style.setProperty("height", h + "px", "important");
                } else {
                    canvas.style.removeProperty( "width");
                    canvas.style.removeProperty("height");
                }
            }
        }
    });
    function _emscripten_cancel_main_loop() {
        Browser.mainLoop.pause();
        Browser.mainLoop.func = null;
    }

    function _emscripten_clear_interval(id) {
        clearInterval(id);
    }
}

```

```

}

function _emscripten_console_error(str) {
    assert(typeof str == 'number');
    console.error(UTF8ToString(str));
}

var JSEvents = {inEventHandler:0,removeAllEventListeners:function() {
    for (var i = JSEvents.eventHandlers.length-1; i >= 0; --i) {
        JSEvents._removeHandler(i);
    }
    JSEvents.eventHandlers = [];
    JSEvents.deferredCalls = [];
},registerRemoveEventListeners:function() {
    if (!JSEvents.removeEventListenersRegistered) {
        __ATEXIT__.push(JSEvents.removeAllEventListeners);
        JSEvents.removeEventListenersRegistered = true;
    }
},deferredCalls:[],deferCall:function(targetFunction, precedence, argsList) {
    function arraysHaveEqualContent(arrA, arrB) {
        if (arrA.length != arrB.length) return false;

        for (var i in arrA) {
            if (arrA[i] != arrB[i]) return false;
        }
        return true;
    }
    // Test if the given call was already queued, and if so, don't add it again.
    for (var i in JSEvents.deferredCalls) {
        var call = JSEvents.deferredCalls[i];
        if (call.targetFunction == targetFunction &&
arraysHaveEqualContent(call.argsList, argsList)) {
            return;
        }
    }
    JSEvents.deferredCalls.push({
        targetFunction: targetFunction,
        precedence: precedence,
        argsList: argsList
    });
}

JSEvents.deferredCalls.sort(function(x,y) { return x.precedence < y.precedence;
});;
},removeDeferredCalls:function(targetFunction) {
    for (var i = 0; i < JSEvents.deferredCalls.length; ++i) {
        if (JSEvents.deferredCalls[i].targetFunction == targetFunction) {
            JSEvents.deferredCalls.splice(i, 1);
            --i;
        }
    }
},canPerformEventHandlerRequests:function() {
    return JSEvents.inEventHandler &&
JSEvents.currentEventHandler.allowsDeferredCalls;
},runDeferredCalls:function() {
    if (!JSEvents.canPerformEventHandlerRequests()) {

```

```

        return;
    }
    for (var i = 0; i < JSEvents.deferredCalls.length; ++i) {
        var call = JSEvents.deferredCalls[i];
        JSEvents.deferredCalls.splice(i, 1);
        --i;
        call.targetFunction.apply(null, call.argsList);
    }
},eventHandlers:[],removeAllHandlersOnTarget:function(target, eventTypeString) {
    for (var i = 0; i < JSEvents.eventHandlers.length; ++i) {
        if (JSEvents.eventHandlers[i].target == target &&
            (!eventTypeString || eventTypeString ==
JSEvents.eventHandlers[i].eventTypeString)) {
            JSEvents._removeHandler(i--);
        }
    }
},_removeHandler:function(i) {
    var h = JSEvents.eventHandlers[i];
    h.target.removeEventListener(h.eventTypeString, h.eventListenerFunc,
h.useCapture);
    JSEvents.eventHandlers.splice(i, 1);
},registerOrRemoveHandler:function(eventHandler) {
    var jsEventHandler = function jsEventHandler(event) {
        // Increment nesting count for the event handler.
        ++JSEvents.inEventHandler;
        JSEvents.currentEventHandler = eventHandler;
        // Process any old deferred calls the user has placed.
        JSEvents.runDeferredCalls();
        // Process the actual event, calls back to user C code handler.
        eventHandler.handlerFunc(event);
        // Process any new deferred calls that were placed right now from this event
handler.
        JSEvents.runDeferredCalls();
        // Out of event handler - restore nesting count.
        --JSEvents.inEventHandler;
    };
    if (eventHandler.callbackfunc) {
        eventHandler.eventListenerFunc = jsEventHandler;
        eventHandler.target.addEventListener(eventHandler.eventTypeString,
jsEventHandler, eventHandler.useCapture);
        JSEvents.eventHandlers.push(eventHandler);
        JSEvents.registerRemoveEventListeners();
    } else {
        for (var i = 0; i < JSEvents.eventHandlers.length; ++i) {
            if (JSEvents.eventHandlers[i].target == eventHandler.target
                && JSEvents.eventHandlers[i].eventTypeString ==
eventHandler.eventTypeString) {
                JSEvents._removeHandler(i--);
            }
        }
    }
},getNodeNameForTarget:function(target) {
    if (!target) return '';
    if (target == window) return '#window';
}

```

```

    if (target == screen) return '#screen';
    return (target && target.nodeName) ? target.nodeName : '';
}, fullscreenEnabled:function() {
    return document.fullscreenEnabled
    // Safari 13.0.3 on macOS Catalina 10.15.1 still ships with prefixed
webkitFullscreenEnabled.
    // TODO: If Safari at some point ships with unprefixed version, update the
version check above.
    || document.webkitFullscreenEnabled
    ;
}};

var currentFullscreenStrategy = {};

function maybeCStringToJsString(cString) {
    // "cString > 2" checks if the input is a number, and isn't of the special
    // values we accept here, EMScripten_EVENT_TARGET_* (which map to 0, 1, 2).
    // In other words, if cString > 2 then it's a pointer to a valid place in
    // memory, and points to a C string.
    return cString > 2 ? UTF8ToString(cString) : cString;
}

var specialHTMLTargets = [0, typeof document != 'undefined' ? document : 0, typeof
window != 'undefined' ? window : 0];
function findEventTarget(target) {
    target = maybeCStringToJsString(target);
    var domElement = specialHTMLTargets[target] || (typeof document != 'undefined' ?
document.querySelector(target) : undefined);
    return domElement;
}
function findCanvasEventTarget(target) { return findEventTarget(target); }
function _emscripten_get_canvas_element_size(target, width, height) {
    var canvas = findCanvasEventTarget(target);
    if (!canvas) return -4;
    HEAP32[((width)>>2)] = canvas.width;
    HEAP32[((height)>>2)] = canvas.height;
}
function getCanvasElementSize(target) {
    return withStackSave(function() {
        var w = stackAlloc(8);
        var h = w + 4;

        var targetInt = stackAlloc(target.id.length+1);
        stringToUTF8(target.id, targetInt, target.id.length+1);
        var ret = _emscripten_get_canvas_element_size(targetInt, w, h);
        var size = [HEAP32[((w)>>2)], HEAP32[((h)>>2)]];
        return size;
    });
}

function _emscripten_set_canvas_element_size(target, width, height) {
    var canvas = findCanvasEventTarget(target);
    if (!canvas) return -4;
    canvas.width = width;
    canvas.height = height;
}

```

```

        return 0;
    }
}

function setCanvasElementSize(target, width, height) {
    if (!target.controlTransferredOffscreen) {
        target.width = width;
        target.height = height;
    } else {
        // This function is being called from high-level JavaScript code instead of
        // asm.js/Wasm,
        // and it needs to synchronously proxy over to another thread, so marshal the
        // string onto the heap to do the call.
        withStackSave(function() {
            var targetInt = stackAlloc(target.id.length+1);
            stringToUTF8(target.id, targetInt, target.id.length+1);
            _emscripten_set_canvas_element_size(targetInt, width, height);
        });
    }
}

function registerRestoreOldStyle(canvas) {
    var canvasSize = getCanvasElementSize(canvas);
    var oldWidth = canvasSize[0];
    var oldHeight = canvasSize[1];
    var oldCssWidth = canvas.style.width;
    var oldCssHeight = canvas.style.height;
    var oldBackgroundColor = canvas.style.backgroundColor; // Chrome reads color from
here.
    var oldDocumentBackgroundColor = document.body.style.backgroundColor; // IE11
reads color from here.
    // Firefox always has black background color.
    var oldPaddingLeft = canvas.style.paddingLeft; // Chrome, FF, Safari
    var oldPaddingRight = canvas.style.paddingRight;
    var oldPaddingTop = canvas.style.paddingTop;
    var oldPaddingBottom = canvas.style.paddingBottom;
    var oldMarginLeft = canvas.style.marginLeft; // IE11
    var oldMarginRight = canvas.style.marginRight;
    var oldMarginTop = canvas.style.marginTop;
    var oldMarginBottom = canvas.style.marginBottom;
    var oldDocumentBodyMargin = document.body.style.margin;
    var oldDocumentOverflow = document.documentElement.style.overflow; // Chrome,
Firefox
    var oldDocumentScroll = document.body.scroll; // IE
    var oldImageRendering = canvas.style.imageRendering;

    function restoreOldStyle() {
        var fullscreenElement = document.fullscreenElement
            || document.webkitFullscreenElement
            || document.msFullscreenElement
            ;
        if (!fullscreenElement) {
            document.removeEventListener('fullscreenchange', restoreOldStyle);

            // Unprefixed Fullscreen API shipped in Chromium 71
            (https://bugs.chromium.org/p/chromium/issues/detail?id=383813)
            // As of Safari 13.0.3 on macOS Catalina 10.15.1 still ships with prefixed
            webkitfullscreenchange. TODO: revisit this check once Safari ships unprefixed version.
        }
    }
}

```

```

document.removeEventListener('webkitfullscreenchange', restoreOldStyle);

setCanvasElementSize(canvas, oldWidth, oldHeight);

canvas.style.width = oldCssWidth;
canvas.style.height = oldCssHeight;
canvas.style.backgroundColor = oldBackgroundColor; // Chrome
// IE11 hack: assigning 'undefined' or an empty string to
document.body.style.backgroundColor has no effect, so first assign back the default
color
// before setting the undefined value. Setting undefined value is also
important, or otherwise we would later treat that as something that the user
// had explicitly set so subsequent fullscreen transitions would not set
background color properly.
if (!oldDocumentBackgroundColor) document.body.style.backgroundColor =
'white';
document.body.style.backgroundColor = oldDocumentBackgroundColor; // IE11
canvas.style.paddingLeft = oldPaddingLeft; // Chrome, FF, Safari
canvas.style.paddingRight = oldPaddingRight;
canvas.style.paddingTop = oldPaddingTop;
canvas.style.paddingBottom = oldPaddingBottom;
canvas.style.marginLeft = oldMarginLeft; // IE11
canvas.style.marginRight = oldMarginRight;
canvas.style.marginTop = oldMarginTop;
canvas.style.marginBottom = oldMarginBottom;
document.body.style.margin = oldDocumentBodyMargin;
document.documentElement.style.overflow = oldDocumentOverflow; // Chrome,
Firefox
document.body.scroll = oldDocumentScroll; // IE
canvas.style.imageRendering = oldImageRendering;
if (canvas.GLctxObject) canvas.GLctxObject.GLctx.viewport(0, 0, oldWidth,
oldHeight);

if (currentFullscreenStrategy.canvasResizedCallback) {
  (function(a1, a2, a3) { return dynCall_iiii.apply(null,
[currentFullscreenStrategy.canvasResizedCallback, a1, a2, a3]); }) (37, 0,
currentFullscreenStrategy.canvasResizedCallbackUserData);
}
}
}
document.addEventListener('fullscreenchange', restoreOldStyle);
// Unprefixed Fullscreen API shipped in Chromium 71
(https://bugs.chromium.org/p/chromium/issues/detail?id=383813)
// As of Safari 13.0.3 on macOS Catalina 10.15.1 still ships with prefixed
webkitfullscreenchange. TODO: revisit this check once Safari ships unprefixed version.
document.addEventListener('webkitfullscreenchange', restoreOldStyle);
return restoreOldStyle;
}

function setLetterbox(element, topBottom, leftRight) {
  // Cannot use margin to specify letterboxes in FF or Chrome, since those ignore
  margins in fullscreen mode.
  element.style.paddingLeft = element.style.paddingRight = leftRight + 'px';
  element.style.paddingTop = element.style.paddingBottom = topBottom + 'px';
}

```

```

function getBoundingClientRect(e) {
    return specialHTMLTargets.indexOf(e) < 0 ? e.getBoundingClientRect() :
{'left':0,'top':0};
}
function _JSEvents_resizeCanvasForFullscreen(target, strategy) {
    var restoreOldStyle = registerRestoreOldStyle(target);
    var cssWidth = strategy.softFullscreen ? innerWidth : screen.width;
    var cssHeight = strategy.softFullscreen ? innerHeight : screen.height;
    var rect = getBoundingClientRect(target);
    var windowedCssWidth = rect.width;
    var windowedCssHeight = rect.height;
    var canvasSize = getCanvasElementSize(target);
    var windowedRttWidth = canvasSize[0];
    var windowedRttHeight = canvasSize[1];

    if (strategy.scaleMode == 3) {
        setLetterbox(target, (cssHeight - windowedCssHeight) / 2, (cssWidth -
windowedCssWidth) / 2);
        cssWidth = windowedCssWidth;
        cssHeight = windowedCssHeight;
    } else if (strategy.scaleMode == 2) {
        if (cssWidth*windowedRttHeight < windowedRttWidth*cssHeight) {
            var desiredCssHeight = windowedRttHeight * cssWidth / windowedRttWidth;
            setLetterbox(target, (cssHeight - desiredCssHeight) / 2, 0);
            cssHeight = desiredCssHeight;
        } else {
            var desiredCssWidth = windowedRttWidth * cssHeight / windowedRttHeight;
            setLetterbox(target, 0, (cssWidth - desiredCssWidth) / 2);
            cssWidth = desiredCssWidth;
        }
    }
}

// If we are adding padding, must choose a background color or otherwise Chrome
will give the
// padding a default white color. Do it only if user has not customized their own
background color.
if (!target.style.backgroundColor) target.style.backgroundColor = 'black';
// IE11 does the same, but requires the color to be set in the document body.
if (!document.body.style.backgroundColor) document.body.style.backgroundColor =
'black'; // IE11
// Firefox always shows black letterboxes independent of style color.

target.style.width = cssWidth + 'px';
target.style.height = cssHeight + 'px';

if (strategy.filteringMode == 1) {
    target.style.imageRendering = 'optimizeSpeed';
    target.style.imageRendering = '-moz-crisp-edges';
    target.style.imageRendering = '-o-crisp-edges';
    target.style.imageRendering = '-webkit-optimize-contrast';
    target.style.imageRendering = 'optimize-contrast';
    target.style.imageRendering = 'crisp-edges';
    target.style.imageRendering = 'pixelated';
}

```

```

var dpiScale = (strategy.canvasResolutionScaleMode == 2) ? devicePixelRatio : 1;
if (strategy.canvasResolutionScaleMode != 0) {
    var newWidth = (cssWidth * dpiScale)|0;
    var newHeight = (cssHeight * dpiScale)|0;
    setCanvasElementSize(target, newWidth, newHeight);
    if (target.GLctxObject) target.GLctxObject.GLctx.viewport(0, 0, newWidth,
newHeight);
}
return restoreOldStyle;
}
function _JSEvents_requestFullscreen(target, strategy) {
// EMSCRIPTEN_FULLSCREEN_SCALE_DEFAULT + EMSCRIPTEN_FULLSCREEN_CANVAS_SCALE_NONE
is a mode where no extra logic is performed to the DOM elements.
if (strategy.scaleMode != 0 || strategy.canvasResolutionScaleMode != 0) {
    _JSEvents_resizeCanvasForFullscreen(target, strategy);
}

if (target.requestFullscreen) {
    target.requestFullscreen();
} else if (target.webkitRequestFullscreen) {
    target.webkitRequestFullscreen(Element.ALLOW_KEYBOARD_INPUT);
} else {
    return JSEvents.fullscreenEnabled() ? -3 : -1;
}

currentFullscreenStrategy = strategy;

if (strategy.canvasResizedCallback) {
    (function(a1, a2, a3) { return dynCall_iiii.apply(null,
[strategy.canvasResizedCallback, a1, a2, a3]); })(37, 0,
strategy.canvasResizedCallbackUserData);
}

return 0;
}
function emscripten_exit_fullscreen() {
if (!JSEvents.fullscreenEnabled()) return -1;
// Make sure no queued up calls will fire after this.
JSEvents.removeDeferredCalls(_JSEvents_requestFullscreen);

var d = specialHTMLTargets[1];
if (d.exitFullscreen) {
    d.fullscreenElement && d.exitFullscreen();
} else if (d.webkitExitFullscreen) {
    d.webkitFullscreenElement && d.webkitExitFullscreen();
} else {
    return -1;
}

return 0;
}

function requestPointerLock(target) {
if (target.requestPointerLock) {
    target.requestPointerLock();
} else if (target.msRequestPointerLock) {

```

```

        target.msRequestPointerLock();
    } else {
        // document.body is known to accept pointer lock, so use that to differentiate
        if the user passed a bad element,
            // or if the whole browser just doesn't support the feature.
            if (document.body.requestPointerLock
                || document.body.msRequestPointerLock
            ) {
                return -3;
            } else {
                return -1;
            }
        }
        return 0;
    }
}

function _emscripten_exit_pointerlock() {
    // Make sure no queued up calls will fire after this.
    JSEvents.removeDeferredCalls(requestPointerLock);

    if (document.exitPointerLock) {
        document.exitPointerLock();
    } else if (document.msExitPointerLock) {
        document.msExitPointerLock();
    } else {
        return -1;
    }
    return 0;
}

function fillFullscreenChangeEventData(eventStruct) {
    var fullscreenElement = document.fullscreenElement ||
document.mozFullScreenElement || document.webkitFullscreenElement ||
document.msFullscreenElement;
    var isFullscreen = !fullscreenElement;
    // Assigning a boolean to HEAP32 with expected type coercion.
    /** @suppress{checkTypes} */
    HEAP32[((eventStruct)>>2)] = isFullscreen;
    HEAP32[((eventStruct)+(4))>>2)] = JSEvents.fullscreenEnabled();
    // If transitioning to fullscreen, report info about the element that is now
    fullscreen.
    // If transitioning to windowed mode, report info about the element that just was
    fullscreen.
    var reportedElement = isFullscreen ? fullscreenElement :
JSEvents.previousFullscreenElement;
    var nodeName = JSEvents.getNodeNameForTarget(reportedElement);
    var id = (reportedElement && reportedElement.id) ? reportedElement.id : '';
    stringToUTF8(nodeName, eventStruct + 8, 128);
    stringToUTF8(id, eventStruct + 136, 128);
    HEAP32[((eventStruct)+(264))>>2)] = reportedElement ?
reportedElement.clientWidth : 0;
    HEAP32[((eventStruct)+(268))>>2)] = reportedElement ?
reportedElement.clientHeight : 0;
    HEAP32[((eventStruct)+(272))>>2)] = screen.width;
    HEAP32[((eventStruct)+(276))>>2)] = screen.height;
    if (isFullscreen) {

```

```

        JSEvents.previousFullscreenElement = fullscreenElement;
    }
}

function _emscripten_get_fullscreen_status(fullscreenStatus) {
    if (!JSEvents.fullscreenEnabled()) return -1;
    fillFullscreenChangeEventData(fullscreenStatus);
    return 0;
}

function fillGamepadEventData(eventStruct, e) {
    HEAPF64[((eventStruct)>>3)] = e.timestamp;
    for (var i = 0; i < e.axes.length; ++i) {
        HEAPF64[((eventStruct+i*8)+(16))>>3] = e.axes[i];
    }
    for (var i = 0; i < e.buttons.length; ++i) {
        if (typeof e.buttons[i] == 'object') {
            HEAPF64[((eventStruct+i*8)+(528))>>3] = e.buttons[i].value;
        } else {
            HEAPF64[((eventStruct+i*8)+(528))>>3] = e.buttons[i];
        }
    }
    for (var i = 0; i < e.buttons.length; ++i) {
        if (typeof e.buttons[i] == 'object') {
            HEAP32[((eventStruct+i*4)+(1040))>>2] = e.buttons[i].pressed;
        } else {
            // Assigning a boolean to HEAP32, that's ok, but Closure would like to warn
            about it:
            /* @suppress {checkTypes} */
            HEAP32[((eventStruct+i*4)+(1040))>>2] = e.buttons[i] == 1;
        }
    }
    HEAP32[((eventStruct)+(1296))>>2] = e.connected;
    HEAP32[((eventStruct)+(1300))>>2] = e.index;
    HEAP32[((eventStruct)+(8))>>2] = e.axes.length;
    HEAP32[((eventStruct)+(12))>>2] = e.buttons.length;
    stringToUTF8(e.id, eventStruct + 1304, 64);
    stringToUTF8(e.mapping, eventStruct + 1368, 64);
}
function _emscripten_get_gamepad_status(index, gamepadState) {
    if (!JSEvents.lastGamepadState) throw 'emscripten_get_gamepad_status() can only
be called after having first called emscripten_sample_gamepad_data() and that function
has returned EMSCRIPTEN_RESULT_SUCCESS!';

    // INVALID_PARAM is returned on a Gamepad index that never was there.
    if (index < 0 || index >= JSEvents.lastGamepadState.length) return -5;

    // NO_DATA is returned on a Gamepad index that was removed.
    // For previously disconnected gamepads there should be an empty slot
    (null/undefined/false) at the index.
    // This is because gamepads must keep their original position in the array.
    // For example, removing the first of two gamepads produces
    [null/undefined/false, gamepad].
    if (!JSEvents.lastGamepadState[index]) return -7;

    fillGamepadEventData(gamepadState, JSEvents.lastGamepadState[index]);
}

```

```

        return 0;
    }

function _emscripten_get_heap_max() {
    // Stay one Wasm page short of 4GB: while e.g. Chrome is able to allocate
    // full 4GB Wasm memories, the size will wrap back to 0 bytes in Wasm side
    // for any code that deals with heap sizes, which would require special
    // casing all heap size related code to treat 0 specially.
    return 2147483648;
}

function _emscripten_get_now_res() { // return resolution of get_now, in nanoseconds
    if (ENVIRONMENT_IS_NODE) {
        return 1; // nanoseconds
    } else
        // Modern environment where performance.now() is supported:
        return 1000; // microseconds (1/1000 of a millisecond)
}

function _emscripten_get_num_gamepads() {
    if (!JSEvents.lastGamepadState) throw 'emscripten_get_num_gamepads() can only be
called after having first called emscripten_sample_gamepad_data() and that function has
returned EMSCRIPTEN_RESULT_SUCCESS!';
    // N.B. Do not call emscripten_get_num_gamepads() unless having first called
    emscripten_sample_gamepad_data(), and that has returned EMSCRIPTEN_RESULT_SUCCESS.
    // Otherwise the following line will throw an exception.
    return JSEvents.lastGamepadState.length;
}

function _emscripten_html5_remove_all_event_listeners() {
    JSEvents.removeAllEventListeners();
}

function _emscripten_is_webgl_context_lost(contextHandle) {
    return !GL.contexts[contextHandle] ||
    GL.contexts[contextHandle].GLctx.isContextLost(); // No context ~> lost context.
}

function reallyNegative(x) {
    return x < 0 || (x === 0 && (1/x) === -Infinity);
}

function convertI32PairToI53(lo, hi) {
    // This function should not be getting called with too large unsigned numbers
    // in high part (if hi >= 0x7FFFFFFF, one should have been calling
    // convertU32PairToI53())
    assert(hi === (hi|0));
    return (lo >>> 0) + hi * 4294967296;
}

function convertU32PairToI53(lo, hi) {
    return (lo >>> 0) + (hi >>> 0) * 4294967296;
}

```

```

function reSign(value, bits) {
    if (value <= 0) {
        return value;
    }
    var half = bits <= 32 ? Math.abs(1 << (bits-1)) // abs is needed if bits == 32
                          : Math.pow(2, bits-1);
    // for huge values, we can hit the precision limit and always get true here.
    // so don't do that but, in general there is no perfect solution here. With
    // 64-bit ints, we get rounding and errors
    // TODO: In i64 mode 1, resign the two parts separately and safely
    if (value >= half && (bits <= 32 || value > half)) {
        // Cannot bitshift half, as it may be at the limit of the bits JS uses in
        // bitshifts
        value = -2*half + value;
    }
    return value;
}

function unSign(value, bits) {
    if (value >= 0) {
        return value;
    }
    // Need some trickery, since if bits == 32, we are right at the limit of the
    // bits JS uses in bitshifts
    return bits <= 32 ? 2*Math.abs(1 << (bits-1)) + value
                      : Math.pow(2, bits)           + value;
}
function formatString(format, varargs) {
;
assert((varargs & 3) === 0);
var textIndex = format;
var argIndex = varargs;
// This must be called before reading a double or i64 vararg. It will bump the
pointer properly.
// It also does an assert on i32 values, so it's nice to call it before all
varargs calls.
function prepVararg(ptr, type) {
    if (type === 'double' || type === 'i64') {
        // move so the load is aligned
        if (ptr & 7) {
            assert((ptr & 7) === 4);
            ptr += 4;
        }
    } else {
        assert((ptr & 3) === 0);
    }
    return ptr;
}
function getNextArg(type) {
    // NOTE: Explicitly ignoring type safety. Otherwise this fails:
    //       int x = 4; printf("%c\n", (char)x);
    var ret;
    argIndex = prepVararg(argIndex, type);
    if (type === 'double') {
        ret = Number(HEAPF64[((argIndex)>>3)]);
    }
}

```

```

        argIndex += 8;
    } else if (type == 'i64') {
        ret = [HEAP32[((argIndex)>>2)],  

               HEAP32[((((argIndex)+(4))>>2)]];
        argIndex += 8;
    } else {
        assert((argIndex & 3) === 0);
        type = 'i32'; // varargs are always i32, i64, or double
        ret = HEAP32[((argIndex)>>2)];
        argIndex += 4;
    }
    return ret;
}

var ret = [];
var curr, next, currArg;
while (1) {
    var startTextIndex = textIndex;
    curr = HEAP8[((textIndex)>>0)];
    if (curr === 0) break;
    next = HEAP8[((textIndex+1)>>0)];
    if (curr === 37) {
        // Handle flags.
        var flagAlwaysSigned = false;
        var flagLeftAlign = false;
        var flagAlternative = false;
        var flagZeroPad = false;
        var flagPadSign = false;
        flagsLoop: while (1) {
            switch (next) {
                case 43:
                    flagAlwaysSigned = true;
                    break;
                case 45:
                    flagLeftAlign = true;
                    break;
                case 35:
                    flagAlternative = true;
                    break;
                case 48:
                    if (flagZeroPad) {
                        break flagsLoop;
                    } else {
                        flagZeroPad = true;
                        break;
                    }
                case 32:
                    flagPadSign = true;
                    break;
                default:
                    break flagsLoop;
            }
            textIndex++;
            next = HEAP8[((textIndex+1)>>0)];
        }
    }
}

```

```

// Handle width.
var width = 0;
if (next == 42) {
    width = getNextArg('i32');
    textIndex++;
    next = HEAP8[((textIndex+1)>>0)];
} else {
    while (next >= 48 && next <= 57) {
        width = width * 10 + (next - 48);
        textIndex++;
        next = HEAP8[((textIndex+1)>>0)];
    }
}

// Handle precision.
var precisionSet = false, precision = -1;
if (next == 46) {
    precision = 0;
    precisionSet = true;
    textIndex++;
    next = HEAP8[((textIndex+1)>>0)];
    if (next == 42) {
        precision = getNextArg('i32');
        textIndex++;
    } else {
        while (1) {
            var precisionChr = HEAP8[((textIndex+1)>>0)];
            if (precisionChr < 48 || precisionChr > 57) break;
            precision = precision * 10 + (precisionChr - 48);
            textIndex++;
        }
    }
    next = HEAP8[((textIndex+1)>>0)];
}
if (precision < 0) {
    precision = 6; // Standard default.
    precisionSet = false;
}

// Handle integer sizes. WARNING: These assume a 32-bit architecture!
var argSize;
switch (String.fromCharCode(next)) {
    case 'h':
        var nextNext = HEAP8[((textIndex+2)>>0)];
        if (nextNext == 104) {
            textIndex++;
            argSize = 1; // char (actually i32 in varargs)
        } else {
            argSize = 2; // short (actually i32 in varargs)
        }
        break;
    case 'l':
        var nextNext = HEAP8[((textIndex+2)>>0)];

```

```

    if (nextNext == 108) {
        textIndex++;
        argSize = 8; // long long
    } else {
        argSize = 4; // long
    }
    break;
case 'L': // long long
case 'q': // int64_t
case 'j': // intmax_t
    argSize = 8;
    break;
case 'z': // size_t
case 't': // ptrdiff_t
case 'I': // signed ptrdiff_t or unsigned size_t
    argSize = 4;
    break;
default:
    argSize = null;
}
if (argSize) textIndex++;
next = HEAP8[((textIndex+1)>>0)];

// Handle type specifier.
switch (String.fromCharCode(next)) {
case 'd': case 'i': case 'u': case 'o': case 'x': case 'X': case 'p': {
    // Integer.
    var signed = next == 100 || next == 105;
    argSize = argSize || 4;
    currArg = getNextArg('i' + (argSize * 8));
    var argText;
    // Flatten i64-1 [low, high] into a (slightly rounded) double
    if (argSize == 8) {
        currArg = next == 117 ? convertU32PairToI53(currArg[0], currArg[1]) :
convertI32PairToI53(currArg[0], currArg[1]);
    }
    // Truncate to requested size.
    if (argSize <= 4) {
        var limit = Math.pow(256, argSize) - 1;
        currArg = (signed ? reSign : unSign)(currArg & limit, argSize * 8);
    }
    // Format the number.
    var currAbsArg = Math.abs(currArg);
    var prefix = '';
    if (next == 100 || next == 105) {
        argText = reSign(currArg, 8 * argSize).toString(10);
    } else if (next == 117) {
        argText = unSign(currArg, 8 * argSize).toString(10);
        currArg = Math.abs(currArg);
    } else if (next == 111) {
        argText = (flagAlternative ? '0' : '') + currAbsArg.toString(8);
    } else if (next == 120 || next == 88) {
        prefix = (flagAlternative && currArg != 0) ? '0x' : '';
        if (currArg < 0) {
            // Represent negative numbers in hex as 2's complement.

```

```

currArg = -currArg;
argText = (currAbsArg - 1).toString(16);
var buffer = [];
for (var i = 0; i < argText.length; i++) {
    buffer.push((0xF - parseInt(argText[i], 16)).toString(16));
}
argText = buffer.join('');
while (argText.length < argSize * 2) argText = 'f' + argText;
} else {
    argText = currAbsArg.toString(16);
}
if (next == 88) {
    prefix = prefix.toUpperCase();
    argText = argText.toUpperCase();
}
} else if (next == 112) {
    if (currAbsArg === 0) {
        argText = '(nil)';
    } else {
        prefix = '0x';
        argText = currAbsArg.toString(16);
    }
}
if (precisionSet) {
    while (argText.length < precision) {
        argText = '0' + argText;
    }
}

// Add sign if needed
if (currArg >= 0) {
    if (flagAlwaysSigned) {
        prefix = '+' + prefix;
    } else if (flagPadSign) {
        prefix = ' ' + prefix;
    }
}

// Move sign to prefix so we zero-pad after the sign
if (argText.charAt(0) == '-') {
    prefix = '-' + prefix;
    argText = argText.substr(1);
}

// Add padding.
while (prefix.length + argText.length < width) {
    if (flagLeftAlign) {
        argText += ' ';
    } else {
        if (flagZeroPad) {
            argText = '0' + argText;
        } else {
            prefix = ' ' + prefix;
        }
    }
}

```

```

}

// Insert the result into the buffer.
argText = prefix + argText;
argText.split('').forEach(function(chr) {
  ret.push(chr.charCodeAt(0));
});
break;
}
case 'f': case 'F': case 'e': case 'E': case 'g': case 'G': {
// Float.
currArg = getNextArg('double');
var argText;
if (isNaN(currArg)) {
  argText = 'nan';
  flagZeroPad = false;
} else if (!isFinite(currArg)) {
  argText = (currArg < 0 ? '-' : '') + 'inf';
  flagZeroPad = false;
} else {
  var isGeneral = false;
  var effectivePrecision = Math.min(precision, 20);

  // Convert g/G to f/F or e/E, as per:
  //
http://pubs.opengroup.org/onlinepubs/9699919799/functions/printf.html
  if (next == 103 || next == 71) {
    isGeneral = true;
    precision = precision || 1;
    var exponent =
      parseInt(currArg.toExponential(effectivePrecision).split('e')[1], 10);
    if (precision > exponent && exponent >= -4) {
      next = ((next == 103) ? 'f' : 'F').charCodeAt(0);
      precision -= exponent + 1;
    } else {
      next = ((next == 103) ? 'e' : 'E').charCodeAt(0);
      precision--;
    }
    effectivePrecision = Math.min(precision, 20);
  }

  if (next == 101 || next == 69) {
    argText = currArg.toExponential(effectivePrecision);
    // Make sure the exponent has at least 2 digits.
    if (/[^eE][-+]\d/.test(argText)) {
      argText = argText.slice(0, -1) + '0' + argText.slice(-1);
    }
  } else if (next == 102 || next == 70) {
    argText = currArg.toFixed(effectivePrecision);
    if (currArg === 0 && reallyNegative(currArg)) {
      argText = '-' + argText;
    }
  }
}

var parts = argText.split('e');
```

```

    if (isGeneral && !flagAlternative) {
        // Discard trailing zeros and periods.
        while (parts[0].length > 1 && parts[0].includes('.') &&
               (parts[0].slice(-1) == '0' || parts[0].slice(-1) == '.')) {
            parts[0] = parts[0].slice(0, -1);
        }
    } else {
        // Make sure we have a period in alternative mode.
        if (flagAlternative && argText.indexOf('.') == -1) parts[0] += '.';
        // Zero pad until required precision.
        while (precision > effectivePrecision++) parts[0] += '0';
    }
    argText = parts[0] + (parts.length > 1 ? 'e' + parts[1] : '');

    // Capitalize 'E' if needed.
    if (next == 69) argText = argText.toUpperCase();

    // Add sign.
    if (currArg >= 0) {
        if (flagAlwaysSigned) {
            argText = '+' + argText;
        } else if (flagPadSign) {
            argText = ' ' + argText;
        }
    }
}

// Add padding.
while (argText.length < width) {
    if (flagLeftAlign) {
        argText += ' ';
    } else {
        if (flagZeroPad && (argText[0] == '-' || argText[0] == '+')) {
            argText = argText[0] + '0' + argText.slice(1);
        } else {
            argText = (flagZeroPad ? '0' : ' ') + argText;
        }
    }
}

// Adjust case.
if (next < 97) argText = argText.toUpperCase();

// Insert the result into the buffer.
argText.split('').forEach(function(chr) {
    ret.push(chr.charCodeAt(0));
});
break;
}

case 's': {
    // String.
    var arg = getNextArg('i8*');
    var argLength = arg ? _strlen(arg) : '(null).length';
    if (precisionSet) argLength = Math.min(argLength, precision);
    if (!flagLeftAlign) {

```

```

        while (argLength < width--) {
            ret.push(32);
        }
    }
    if (arg) {
        for (var i = 0; i < argLength; i++) {
            ret.push(HEAPU8[((arg++)>>0)]);
        }
    } else {
        ret = ret.concat(intArrayFromString('(null)').substr(0, argLength),
true));
    }
    if (flagLeftAlign) {
        while (argLength < width--) {
            ret.push(32);
        }
    }
    break;
}
case 'c': {
    // Character.
    if (flagLeftAlign) ret.push(getNextArg('i8'));
    while (--width > 0) {
        ret.push(32);
    }
    if (!flagLeftAlign) ret.push(getNextArg('i8'));
    break;
}
case 'n': {
    // Write the length written so far to the next parameter.
    var ptr = getNextArg('i32*');
    HEAP32[((ptr)>>2)] = ret.length;
    break;
}
case '%': {
    // Literal percent sign.
    ret.push(curr);
    break;
}
default: {
    // Unknown specifiers remain untouched.
    for (var i = startTextIndex; i < textIndex + 2; i++) {
        ret.push(HEAP8[((i)>>0)]);
    }
}
textIndex += 2;
// TODO: Support a/A (hex float) and m (last error) specifiers.
// TODO: Support %1${specifier} for arg selection.
} else {
    ret.push(curr);
    textIndex += 1;
}
}
return ret;

```

```

}

function traverseStack(args) {
    if (!args || !args.callee || !args.callee.name) {
        return [null, '', ''];
    }

    var funstr = args.callee.toString();
    var funcname = args.callee.name;
    var str = '(';
    var first = true;
    for (var i in args) {
        var a = args[i];
        if (!first) {
            str += ", ";
        }
        first = false;
        if (typeof a == 'number' || typeof a == 'string') {
            str += a;
        } else {
            str += '(' + typeof a + ')';
        }
    }
    str += ')';
    var caller = args.callee.caller;
    args = caller ? caller.arguments : [];
    if (first)
        str = '';
    return [args, funcname, str];
}
/** @param {number=} flags */
function _emscripten_get_callstack_js(flags) {
    var callstack = jsStackTrace();

    // Find the symbols in the callstack that corresponds to the functions that
    // report callstack information, and remove everything up to these from the output.
    var iThisFunc = callstack.lastIndexOf('_emscripten_log');
    var iThisFunc2 = callstack.lastIndexOf('_emscripten_get_callstack');
    var iNextLine = callstack.indexOf('\n', Math.max(iThisFunc, iThisFunc2))+1;
    callstack = callstack.slice(iNextLine);

    if (flags & 32) {
        warnOnce('EM_LOG_DEMANGLE is deprecated; ignoring');
    }

    // If user requested to see the original source stack, but no source map
    // information is available, just fall back to showing the JS stack.
    if (flags & 8 && typeof emscripten_source_map == 'undefined') {
        warnOnce('Source map information is not available, emscripten_log with
EM_LOG_C_STACK will be ignored. Build with "--pre-js $EMSCRIPTEN/src/emscripten-source-
map.min.js" linker flag to add source map loading to code.');
        flags ^= 8;
        flags |= 16;
    }
}

```

```

var stack_args = null;
if (flags & 128) {
    // To get the actual parameters to the functions, traverse the stack via the
unfortunately deprecated 'arguments.callee' method, if it works:
    stack_args = traverseStack(arguments);
    while (stack_args[1].includes('_emscripten_'))
        stack_args = traverseStack(stack_args[0]);
}

// Process all lines:
var lines = callstack.split('\n');
callstack = '';
var newFirefoxRe = new RegExp('\\s*(.*?)@(.*):([0-9]+):([0-9]+)'); // New FF30
with column info: extract components of form '
Object._main@http://server.com:4324:12'
var firefoxRe = new RegExp('\\s*(.*?)(.*)(:(.*))?' ); // Old FF without
column info: extract components of form '          Object._main@http://server.com:4324'
var chromeRe = new RegExp('\\sat (.*?) \\\\((.*):(.*):(.*)\\\\)'); // Extract
components of form '      at Object._main (http://server.com/file.html:4324:12)'

for (var l in lines) {
    var line = lines[l];

    var symbolName = '';
    var file = '';
    var lineno = 0;
    var column = 0;

    var parts = chromeRe.exec(line);
    if (parts && parts.length == 5) {
        symbolName = parts[1];
        file = parts[2];
        lineno = parts[3];
        column = parts[4];
    } else {
        parts = newFirefoxRe.exec(line);
        if (!parts) parts = firefoxRe.exec(line);
        if (parts && parts.length >= 4) {
            symbolName = parts[1];
            file = parts[2];
            lineno = parts[3];
            column = parts[4]|0; // Old Firefox doesn't carry column information, but
in new FF30, it is present. See https://bugzilla.mozilla.org/show_bug.cgi?id=762556
        } else {
            // Was not able to extract this line for demangling/sourcemap purposes.
Output it as-is.
            callstack += line + '\n';
            continue;
        }
    }

    var haveSourceMap = false;

    if (flags & 8) {

```

```

        var orig = emscripten_source_map.originalPositionFor({line: lineno, column:
column});
        haveSourceMap = (orig && orig.source);
        if (haveSourceMap) {
            if (flags & 64) {
                orig.source = orig.source.substring(orig.source.replace(/\//g,
"/").lastIndexOf('/')+1);
            }
            callstack += '      at ' + symbolName + ' (' + orig.source + ':' + orig.line
+ ':' + orig.column + ')\n';
        }
    }
    if ((flags & 16) || !haveSourceMap) {
        if (flags & 64) {
            file = file.substring(file.replace(/\//g, "/").lastIndexOf('/')+1);
        }
        callstack += (haveSourceMap ? ('      = ' + symbolName) : ('      at ' +
symbolName)) + ' (' + file + ':' + lineno + ':' + column + ')\n';
    }

    // If we are still keeping track with the callstack by traversing via
'argumentscallee', print the function parameters as well.
    if (flags & 128 && stack_args[0]) {
        if (stack_args[1] == symbolName && stack_args[2].length > 0) {
            callstack = callstack.replace(/\s+$/, '');
            callstack += ' with values: ' + stack_args[1] + stack_args[2] + '\n';
        }
        stack_args = traverseStack(stack_args[0]);
    }
}
// Trim extra whitespace at the end of the output.
callstack = callstack.replace(/\s+$/, '');
return callstack;
}

function _emscripten_log_js(flags, str) {
    if (flags & 24) {
        str = str.replace(/\s+$/, ''); // Ensure the message and the callstack are
joined cleanly with exactly one newline.
        str += (str.length > 0 ? '\n' : '') + _emscripten_get_callstack_js(flags);
    }

    if (flags & 1) {
        if (flags & 4) {
            console.error(str);
        } else if (flags & 2) {
            console.warn(str);
        } else if (flags & 512) {
            console.info(str);
        } else if (flags & 256) {
            console.debug(str);
        } else {
            console.log(str);
        }
    } else if (flags & 6) {
        err(str);
    } else {

```

```

        out(str);
    }
}

function _emscripten_log(flags, format, varargs) {
    var result = formatString(format, varargs);
    var str = UTF8ArrayToString(result, 0);
    _emscripten_log_js(flags, str);
}

function _emscripten_memcpy_big(dest, src, num) {
    HEAPU8.copyWithin(dest, src, src + num);
}

function doRequestFullscreen(target, strategy) {
    if (!JSEvents.fullscreenEnabled()) return -1;
    target = findEventTarget(target);
    if (!target) return -4;

    if (!target.requestFullscreen
        && !target.webkitRequestFullscreen
    ) {
        return -3;
    }

    var canPerformRequests = JSEvents.canPerformEventHandlerRequests();

    // Queue this function call if we're not currently in an event handler and the
    user saw it appropriate to do so.
    if (!canPerformRequests) {
        if (strategy.deferUntilInEventHandler) {
            JSEvents.deferCall(_JSEvents_requestFullscreen, 1 /* priority over pointer
lock */, [target, strategy]);
            return 1;
        } else {
            return -2;
        }
    }

    return _JSEvents_requestFullscreen(target, strategy);
}

function _emscripten_request_fullscreen(target, deferUntilInEventHandler) {
    var strategy = {
        // These options perform no added logic, but just bare request fullscreen.
        scaleMode: 0,
        canvasResolutionScaleMode: 0,
        filteringMode: 0,
        deferUntilInEventHandler: deferUntilInEventHandler,
        canvasResizedCallbackTargetThread: 2
    };
    return doRequestFullscreen(target, strategy);
}

function _emscripten_request_pointerlock(target, deferUntilInEventHandler) {
    target = findEventTarget(target);
    if (!target) return -4;
}
```

```

if (!target.requestPointerLock
    && !target.msRequestPointerLock
) {
    return -1;
}

var canPerformRequests = JSEvents.canPerformEventHandlerRequests();

// Queue this function call if we're not currently in an event handler and the
// user saw it appropriate to do so.
if (!canPerformRequests) {
    if (deferUntilInEventHandler) {
        JSEvents.deferCall(requestPointerLock, 2 /* priority below fullscreen */,
[target]);
        return 1;
    } else {
        return -2;
    }
}

return requestPointerLock(target);
}

function emscripten_realloc_buffer(size) {
    try {
        // round size grow request up to wasm page size (fixed 64KB per spec)
        wasmMemory.grow((size - buffer.byteLength + 65535) >>> 16); // .grow() takes a
delta compared to the previous size
        updateGlobalBufferAndViews(wasmMemory.buffer);
        return 1 /*success*/;
    } catch(e) {
        err('emscripten_realloc_buffer: Attempted to grow heap from ' +
buffer.byteLength + ' bytes to ' + size + ' bytes, but got error: ' + e);
    }
    // implicit 0 return to save code size (caller will cast "undefined" into 0
    // anyhow)
}
function emscripten_resize_heap(requestedSize) {
    var oldSize = HEAPU8.length;
    requestedSize = requestedSize >>> 0;
    // With multithreaded builds, races can happen (another thread might increase the
size
    // in between), so return a failure, and let the caller retry.
    assert(requestedSize > oldSize);

    // Memory resize rules:
    // 1. Always increase heap size to at least the requested size, rounded up
    //     to next page multiple.
    // 2a. If MEMORY_GROWTH_LINEAR_STEP == -1, excessively resize the heap
    //     geometrically: increase the heap size according to
    //     MEMORY_GROWTH_GEOMETRIC_STEP factor (default +20%), At most
    //     overreserve by MEMORY_GROWTH_GEOMETRIC_CAP bytes (default 96MB).
    // 2b. If MEMORY_GROWTH_LINEAR_STEP != -1, excessively resize the heap
    //     linearly: increase the heap size by at least
    //     MEMORY_GROWTH_LINEAR_STEP bytes.
}

```

```

// 3. Max size for the heap is capped at 2048MB-WASM_PAGE_SIZE, or by
//      MAXIMUM_MEMORY, or by ASAN limit, depending on which is smallest
// 4. If we were unable to allocate as much memory, it may be due to
//      over-eager decision to excessively reserve due to (3) above.
//      Hence if an allocation fails, cut down on the amount of excess
//      growth, in an attempt to succeed to perform a smaller allocation.

// A limit is set for how much we can grow. We should not exceed that
// (the wasm binary specifies it, so if we tried, we'd fail anyhow).
var maxHeapSize = emscripten_get_heap_max();
if (requestedSize > maxHeapSize) {
    err('Cannot enlarge memory, asked to go up to ' + requestedSize + ' bytes, but
the limit is ' + maxHeapSize + ' bytes!');
    return false;
}

let alignUp = (x, multiple) => x + (multiple - x % multiple) % multiple;

// Loop through potential heap size increases. If we attempt a too eager
// reservation that fails, cut down on the attempted size and reserve a
// smaller bump instead. (max 3 times, chosen somewhat arbitrarily)
for (var cutDown = 1; cutDown <= 4; cutDown *= 2) {
    var overGrownHeapSize = oldSize * (1 + 0.2 / cutDown); // ensure geometric
growth
    // but limit overreserving (default to capping at +96MB overgrowth at most)
    overGrownHeapSize = Math.min(overGrownHeapSize, requestedSize + 100663296);

    var newSize = Math.min(maxHeapSize, alignUp(Math.max(requestedSize,
overGrownHeapSize), 65536));

    var replacement = emscripten_realloc_buffer(newSize);
    if (replacement) {

        return true;
    }
}
err('Failed to grow the heap from ' + oldSize + ' bytes to ' + newSize + ' bytes,
not enough memory!');
return false;
}

function _emscripten_sample_gamepad_data() {
    return (JSEvents.lastGamepadState = (navigator.getGamepads ?
navigator.getGamepads() : (navigator.webkitGetGamepads ? navigator.webkitGetGamepads()
: null)))
    ? 0 : -1;
}

function registerFocusEventCallback(target, userData, useCapture, callbackfunc,
eventTypeId, eventTypeString, targetThread) {
    if (!JSEvents.focusEvent) JSEvents.focusEvent = _malloc( 256 );

    var focusEventHandlerFunc = function(ev) {
        var e = ev || event;

        var nodeName = JSEvents.getNodeNameForTarget(e.target);

```

```

var id = e.target.id ? e.target.id : '';

var focusEvent = JSEvents.focusEvent;
stringToUTF8(nodeName, focusEvent + 0, 128);
stringToUTF8(id, focusEvent + 128, 128);

if ((function(a1, a2, a3) { return dynCall_iiii.apply(null, [callbackfunc, a1,
a2, a3]); }) (eventId, focusEvent, userData)) e.preventDefault();
};

var eventHandler = {
    target: findEventTarget(target),
    eventTypeString: eventTypeString,
    callbackfunc: callbackfunc,
    handlerFunc: focusEventHandlerFunc,
    useCapture: useCapture
};
JSEvents.registerOrRemoveHandler(eventHandler);
}

function _emscripten_set_blur_callback_on_thread(target, userData, useCapture,
callbackfunc, targetThread) {
    registerFocusEventCallback(target, userData, useCapture, callbackfunc, 12,
"blur", targetThread);
    return 0;
}

function _emscripten_set_focus_callback_on_thread(target, userData, useCapture,
callbackfunc, targetThread) {
    registerFocusEventCallback(target, userData, useCapture, callbackfunc, 13,
"focus", targetThread);
    return 0;
}

function registerFullscreenChangeEventCallback(target, userData, useCapture,
callbackfunc, eventId, eventTypeString, targetThread) {
    if (!JSEvents.fullscreenChangeEvent) JSEvents.fullscreenChangeEvent = _malloc(
280 );

    var fullscreenChangeEventhandlerFunc = function(ev) {
        var e = ev || event;

        var fullscreenChangeEvent = JSEvents.fullscreenChangeEvent;
        fillFullscreenChangeEventData(fullscreenChangeEvent);

        if ((function(a1, a2, a3) { return dynCall_iiii.apply(null, [callbackfunc, a1,
a2, a3]); }) (eventId, fullscreenChangeEvent, userData)) e.preventDefault();
    };

    var eventHandler = {
        target: target,
        eventTypeString: eventTypeString,
        callbackfunc: callbackfunc,
        handlerFunc: fullscreenChangeEventhandlerFunc,
    };
}

```

```

        useCapture: useCapture
    };
    JSEvents.registerOrRemoveHandler(eventHandler);
}
function _emscripten_set_fullscreenchange_callback_on_thread(target, userData,
useCapture, callbackfunc, targetThread) {
    if (!JSEvents.fullscreenEnabled()) return -1;
    target = findEventTarget(target);
    if (!target) return -4;
    registerFullscreenChangeEventCallback(target, userData, useCapture, callbackfunc,
19, "fullscreenchange", targetThread);

    // Unprefixed Fullscreen API shipped in Chromium 71
    // (https://bugs.chromium.org/p/chromium/issues/detail?id=383813)
    // As of Safari 13.0.3 on macOS Catalina 10.15.1 still ships with prefixed
    webkitfullscreenchange. TODO: revisit this check once Safari ships unprefixed version.
    registerFullscreenChangeEventCallback(target, userData, useCapture, callbackfunc,
19, "webkitfullscreenchange", targetThread);

    return 0;
}

function registerGamepadEventCallback(target, userData, useCapture, callbackfunc,
eventTypeId, eventTypeString, targetThread) {
    if (!JSEvents.gamepadEvent) JSEvents.gamepadEvent = _malloc( 1432 );

    var gamepadEventHandlerFunc = function(ev) {
        var e = ev || event;

        var gamepadEvent = JSEvents.gamepadEvent;
        fillGamepadEventData(gamepadEvent, e["gamepad"]);

        if ((function(a1, a2, a3) { return dynCall_iии.apply(null, [callbackfunc, a1,
a2, a3]); })(eventTypeId, gamepadEvent, userData)) e.preventDefault();
    };

    var eventHandler = {
        target: findEventTarget(target),
        allowsDeferredCalls: true,
        eventTypeString: eventTypeString,
        callbackfunc: callbackfunc,
        handlerFunc: gamepadEventHandlerFunc,
        useCapture: useCapture
    };
    JSEvents.registerOrRemoveHandler(eventHandler);
}
function _emscripten_set_gamepadconnected_callback_on_thread(userData, useCapture,
callbackfunc, targetThread) {
    if (!navigator.getGamepads && !navigator.webkitGetGamepads) return -1;
    registerGamepadEventCallback(2, userData, useCapture, callbackfunc, 26,
"gamepadconnected", targetThread);
    return 0;
}

```

```

function _emscripten_set_gamepaddisconnected_callback_on_thread(userData, useCapture,
callbackfunc, targetThread) {
    if (!navigator.getGamepads && !navigator.webkitGetGamepads) return -1;
    registerGamepadEventCallback(2, userData, useCapture, callbackfunc, 27,
"gamepaddisconnected", targetThread);
    return 0;
}

function _emscripten_set_interval(cb, msecs, userData) {

    return setInterval(function() {
        callUserCallback(function() {
            (function(a1) { dynCall_vii.apply(null, [cb, a1]); })(userData)
        });
    }, msecs);
}

function registerKeyEventCallback(target, userData, useCapture, callbackfunc,
eventTypeId, eventTypeString, targetThread) {
    if (!JSEvents.keyEvent) JSEvents.keyEvent = _malloc( 176 );

    var keyEventHandlerFunc = function(e) {
        assert(e);

        var keyEventData = JSEvents.keyEvent;
        HEAPF64[((keyEventData)>>3)] = e.timestamp;

        var idx = keyEventData >> 2;

        HEAP32[idx + 2] = e.location;
        HEAP32[idx + 3] = e.ctrlKey;
        HEAP32[idx + 4] = e.shiftKey;
        HEAP32[idx + 5] = e.altKey;
        HEAP32[idx + 6] = e.metaKey;
        HEAP32[idx + 7] = e.repeat;
        HEAP32[idx + 8] = e.charCode;
        HEAP32[idx + 9] = e.keyCode;
        HEAP32[idx + 10] = e.which;
        stringToUTF8(e.key || '', keyEventData + 44, 32);
        stringToUTF8(e.code || '', keyEventData + 76, 32);
        stringToUTF8(e.char || '', keyEventData + 108, 32);
        stringToUTF8(e.locale || '', keyEventData + 140, 32);

        if ((function(a1, a2, a3) { return dynCall_iiii.apply(null, [callbackfunc, a1,
a2, a3]); })(eventTypeId, keyEventData, userData)) e.preventDefault();
    };

    var eventHandler = {
        target: findEventTarget(target),
        allowsDeferredCalls: true,
        eventTypeString: eventTypeString,
        callbackfunc: callbackfunc,
        handlerFunc: keyEventHandlerFunc,
        useCapture: useCapture
    };
    JSEvents.registerOrRemoveHandler(eventHandler);
}

```

```

        }

    function _emscripten_set_keydown_callback_on_thread(target, userData, useCapture,
callbackfunc, targetThread) {
    registerKeyEventCallback(target, userData, useCapture, callbackfunc, 2,
"keydown", targetThread);
    return 0;
}

function _emscripten_set_keypress_callback_on_thread(target, userData, useCapture,
callbackfunc, targetThread) {
    registerKeyEventCallback(target, userData, useCapture, callbackfunc, 1,
"keypress", targetThread);
    return 0;
}

function _emscripten_set_keyup_callback_on_thread(target, userData, useCapture,
callbackfunc, targetThread) {
    registerKeyEventCallback(target, userData, useCapture, callbackfunc, 3, "keyup",
targetThread);
    return 0;
}

function _emscripten_set_main_loop(func, fps, simulateInfiniteLoop) {
    var browserIterationFunc = (function() { dynCall_v.call(null, func); });
    setMainLoop(browserIterationFunc, fps, simulateInfiniteLoop);
}

function fillMouseEventData(eventStruct, e, target) {
    assert(eventStruct % 4 == 0);
    HEAPF64[((eventStruct)>>3)] = e.timeStamp;
    var idx = eventStruct >> 2;
    HEAP32[idx + 2] = e.screenX;
    HEAP32[idx + 3] = e.screenY;
    HEAP32[idx + 4] = e.clientX;
    HEAP32[idx + 5] = e.clientY;
    HEAP32[idx + 6] = e.ctrlKey;
    HEAP32[idx + 7] = e.shiftKey;
    HEAP32[idx + 8] = e.altKey;
    HEAP32[idx + 9] = e.metaKey;
    HEAP16[idx*2 + 20] = e.button;
    HEAP16[idx*2 + 21] = e.buttons;

    HEAP32[idx + 11] = e["movementX"]
    ;

    HEAP32[idx + 12] = e["movementY"]
    ;

    var rect = getBoundingClientRect(target);
    HEAP32[idx + 13] = e.clientX - rect.left;
    HEAP32[idx + 14] = e.clientY - rect.top;
}

```

```

function registerMouseEventCallback(target, userData, useCapture, callbackfunc,
eventId, eventTypeString, targetThread) {
    if (!JSEvents.mouseEvent) JSEvents.mousePosition = _malloc( 72 );
    target = findEventTarget(target);

    var mouseEventHandlerFunc = function(ev) {
        var e = ev || event;

        // TODO: Make this access thread safe, or this could update live while app is
        reading it.
        fillMouseEventData(JSEvents.mousePosition, e, target);

        if ((function(a1, a2, a3) { return dynCall_iiii.apply(null, [callbackfunc, a1,
a2, a3]); })(eventId, JSEvents.mousePosition, userData)) e.preventDefault();
    };

    var eventHandler = {
        target: target,
        allowsDeferredCalls: eventTypeString != 'mousemove' && eventTypeString != 'mouseenter' && eventTypeString != 'mouseleave', // Mouse move events do not allow
fullscreen/pointer lock requests to be handled in them!
        eventTypeString: eventTypeString,
        callbackfunc: callbackfunc,
        handlerFunc: mouseEventHandlerFunc,
        useCapture: useCapture
    };
    JSEvents.registerOrRemoveHandler(eventHandler);
}

function _emscripten_set_mousedown_callback_on_thread(target, userData, useCapture,
callbackfunc, targetThread) {
    registerMouseEventCallback(target, userData, useCapture, callbackfunc, 5,
"mousedown", targetThread);
    return 0;
}

function _emscripten_set_mousemove_callback_on_thread(target, userData, useCapture,
callbackfunc, targetThread) {
    registerMouseEventCallback(target, userData, useCapture, callbackfunc, 8,
"mousemove", targetThread);
    return 0;
}

function _emscripten_set_mouseup_callback_on_thread(target, userData, useCapture,
callbackfunc, targetThread) {
    registerMouseEventCallback(target, userData, useCapture, callbackfunc, 6,
"mouseup", targetThread);
    return 0;
}

function fillPointerlockChangeEventData(eventStruct) {
    var pointerLockElement = document.pointerLockElement ||
document.mozPointerLockElement || document.webkitPointerLockElement ||
document.msPointerLockElement;
    var isPointerlocked = !!pointerLockElement;
    // Assigning a boolean to HEAP32 with expected type coercion.
    /** @suppress{checkTypes} */
    HEAP32[((eventStruct)>>2)] = isPointerlocked;
}

```

```

var nodeName = JSEvents.getNodeNameForTarget(pointerLockElement);
var id = (pointerLockElement && pointerLockElement.id) ? pointerLockElement.id :
'';
stringToUTF8(nodeName, eventStruct + 4, 128);
stringToUTF8(id, eventStruct + 132, 128);
}
function registerPointerlockChangeEventCallback(target, userData, useCapture,
callbackfunc, eventId, eventString, targetThread) {
    if (!JSEvents.pointerlockChangeEvent) JSEvents.pointerlockChangeEvent = _malloc(
260 );
    var pointerlockChangeEventHandlerFunc = function(ev) {
        var e = ev || event;

        var pointerlockChangeEvent = JSEvents.pointerlockChangeEvent;
        fillPointerlockEventData(pointerlockChangeEvent);

        if ((function(a1, a2, a3) { return dynCall_iiii.apply(null, [callbackfunc, a1,
a2, a3]); }) (eventId, pointerlockChangeEvent, userData)) e.preventDefault();
    };

    var eventHandler = {
        target: target,
        eventString: eventString,
        callbackfunc: callbackfunc,
        handlerFunc: pointerlockChangeEventHandlerFunc,
        useCapture: useCapture
    };
    JSEvents.registerOrRemoveHandler(eventHandler);
}
/** @suppress {missingProperties} */
function _emscripten_set_pointerlockchange_callback_on_thread(target, userData,
useCapture, callbackfunc, targetThread) {
    // TODO: Currently not supported in pthreads or in --proxy-to-worker mode. (In
pthreads mode, document object is not defined)
    if (!document || !document.body || (!document.body.requestPointerLock &&
!document.body.mozRequestPointerLock && !document.body.webkitRequestPointerLock &&
!document.body.msRequestPointerLock)) {
        return -1;
    }

    target = findEventTarget(target);
    if (!target) return -4;
    registerPointerlockChangeEventCallback(target, userData, useCapture,
callbackfunc, 20, "pointerlockchange", targetThread);
    registerPointerlockChangeEventCallback(target, userData, useCapture,
callbackfunc, 20, "mozpointerlockchange", targetThread);
    registerPointerlockChangeEventCallback(target, userData, useCapture,
callbackfunc, 20, "webkitpointerlockchange", targetThread);
    registerPointerlockChangeEventCallback(target, userData, useCapture,
callbackfunc, 20, "mspointerlockchange", targetThread);
    return 0;
}

```

```

function registerTouchEventCallback(target, userData, useCapture, callbackfunc,
eventTypeId, event.TypeString, targetThread) {
    if (!JSEvents.touchEvent) JSEvents.touchEvent = _malloc( 1696 );

    target = findEventTarget(target);

    var touchEventHandlerFunc = function(e) {
        assert(e);
        var t, touches = {}, et = e.touches;
        // To ease marshalling different kinds of touches that browser reports (all
        touches are listed in e.touches,
        // only changed touches in e.changedTouches, and touches on target at
        a.targetTouches), mark a boolean in
        // each Touch object so that we can later loop only once over all touches we
        see to marshall over to Wasm.

        for (var i = 0; i < et.length; ++i) {
            t = et[i];
            // Browser might recycle the generated Touch objects between each frame
            (Firefox on Android), so reset any
            // changed/target states we may have set from previous frame.
            t.isChanged = t.onTarget = 0;
            touches[t.identifier] = t;
        }

        // Mark which touches are part of the changedTouches list.
        for (var i = 0; i < e.changedTouches.length; ++i) {
            t = e.changedTouches[i];
            t.isChanged = 1;
            touches[t.identifier] = t;
        }
        // Mark which touches are part of the targetTouches list.
        for (var i = 0; i < e.targetTouches.length; ++i) {
            touches[e.targetTouches[i].identifier].onTarget = 1;
        }

        var touchEvent = JSEvents.touchEvent;
        var idx = touchEvent>>2; // Pre-shift the ptr to index to HEAP32 to save code
size
        HEAP32[idx + 3] = e.ctrlKey;
        HEAP32[idx + 4] = e.shiftKey;
        HEAP32[idx + 5] = e.altKey;
        HEAP32[idx + 6] = e.metaKey;
        idx += 7; // Advance to the start of the touch array.
        var targetRect = getBoundingClientRect(target);
        var numTouches = 0;
        for (var i in touches) {
            var t = touches[i];
            HEAP32[idx + 0] = t.identifier;
            HEAP32[idx + 1] = t.screenX;
            HEAP32[idx + 2] = t.screenY;
            HEAP32[idx + 3] = t.clientX;
            HEAP32[idx + 4] = t.clientY;
            HEAP32[idx + 5] = t.pageX;
            HEAP32[idx + 6] = t.pageY;
            HEAP32[idx + 7] = t.isChanged;
        }
    }
}

```

```

    HEAP32[idx + 8] = t.onTarget;
    HEAP32[idx + 9] = t.clientX - targetRect.left;
    HEAP32[idx + 10] = t.clientY - targetRect.top;

    idx += 13;

    if (++numTouches > 31) {
        break;
    }
}
HEAP32[((touchEvent)+(8))>>2] = numTouches;

if ((function(a1, a2, a3) { return dynCall_iiii.apply(null, [callbackfunc, a1,
a2, a3]); })(eventId, touchEvent, userData)) e.preventDefault();
};

var eventHandler = {
    target: target,
    allowsDeferredCalls: eventTypeString == 'touchstart' || eventTypeString ==
'touchend',
    eventTypeString: eventTypeString,
    callbackfunc: callbackfunc,
    handlerFunc: touchEventHandlerFunc,
    useCapture: useCapture
};
JSEvents.registerOrRemoveHandler(eventHandler);
}

function _emscripten_set_touchcancel_callback_on_thread(target, userData, useCapture,
callbackfunc, targetThread) {
    registerTouchEventCallback(target, userData, useCapture, callbackfunc, 25,
"touchcancel", targetThread);
    return 0;
}

function _emscripten_set_touchend_callback_on_thread(target, userData, useCapture,
callbackfunc, targetThread) {
    registerTouchEventCallback(target, userData, useCapture, callbackfunc, 23,
"touchend", targetThread);
    return 0;
}

function _emscripten_set_touchmove_callback_on_thread(target, userData, useCapture,
callbackfunc, targetThread) {
    registerTouchEventCallback(target, userData, useCapture, callbackfunc, 24,
"touchmove", targetThread);
    return 0;
}

function _emscripten_set_touchstart_callback_on_thread(target, userData, useCapture,
callbackfunc, targetThread) {
    registerTouchEventCallback(target, userData, useCapture, callbackfunc, 22,
"touchstart", targetThread);
    return 0;
}

```

```

function registerWheelEventCallback(target, userData, useCapture, callbackfunc,
eventId, eventTypeString, targetThread) {
    if (!JSEvents.wheelEvent) JSEvents.wheelEvent = _malloc( 104 );

    // The DOM Level 3 events spec event 'wheel'
    var wheelHandlerFunc = function(ev) {
        var e = ev || event;
        var wheelEvent = JSEvents.wheelEvent;
        fillMouseEventData(wheelEvent, e, target);
        HEAPF64[((wheelEvent)+(72))>>3] = e["deltaX"];
        HEAPF64[((wheelEvent)+(80))>>3] = e["deltaY"];
        HEAPF64[((wheelEvent)+(88))>>3] = e["deltaZ"];
        HEAP32[((wheelEvent)+(96))>>2] = e["deltaMode"];
        if ((function(a1, a2, a3) { return dynCall_iiii.apply(null, [callbackfunc, a1,
a2, a3]); })(eventId, wheelEvent, userData)) e.preventDefault();
    };

    var eventHandler = {
        target: target,
        allowsDeferredCalls: true,
        eventTypeString: eventTypeString,
        callbackfunc: callbackfunc,
        handlerFunc: wheelHandlerFunc,
        useCapture: useCapture
    };
    JSEvents.registerOrRemoveHandler(eventHandler);
}

function _emscripten_set_wheel_callback_on_thread(target, userData, useCapture,
callbackfunc, targetThread) {
    target = findEventTarget(target);
    if (typeof target.onwheel != 'undefined') {
        registerWheelEventCallback(target, userData, useCapture, callbackfunc, 9,
"wheel", targetThread);
        return 0;
    } else {
        return -1;
    }
}

function __webgl_enable_ANGLE_instanced_arrays(ctx) {
    // Extension available in WebGL 1 from Firefox 26 and Google Chrome 30 onwards.
    Core feature in WebGL 2.
    var ext = ctx.getExtension('ANGLE_instanced_arrays');
    if (ext) {
        ctx['vertexAttribDivisor'] = function(index, divisor) {
ext['vertexAttribDivisorANGLE'](index, divisor); };
        ctx['drawArraysInstanced'] = function(mode, first, count, primcount) {
ext['drawArraysInstancedANGLE'](mode, first, count, primcount); };
        ctx['drawElementsInstanced'] = function(mode, count, type, indices, primcount)
{ ext['drawElementsInstancedANGLE'](mode, count, type, indices, primcount); };
        return 1;
    }
}

function __webgl_enable_OES_vertex_array_object(ctx) {

```

```

// Extension available in WebGL 1 from Firefox 25 and WebKit 536.28/desktop
Safari 6.0.3 onwards. Core feature in WebGL 2.
var ext = ctx.getExtension('OES_vertex_array_object');
if (ext) {
    ctx['createVertexArray'] = function() { return ext['createVertexArrayOES'](); };
}
ctx['deleteVertexArray'] = function(vao) { ext['deleteVertexArrayOES'](vao); };
ctx['bindVertexArray'] = function(vao) { ext['bindVertexArrayOES'](vao); };
ctx['isVertexArray'] = function(vao) { return ext['isVertexArrayOES'](vao); };
return 1;
}

function __webgl_enable_WEBGL_draw_buffers(ctx) {
    // Extension available in WebGL 1 from Firefox 28 onwards. Core feature in WebGL
2.
    var ext = ctx.getExtension('WEBGL_draw_buffers');
    if (ext) {
        ctx['drawBuffers'] = function(n, bufs) { ext['drawBuffersWEBGL'](n, bufs); };
        return 1;
    }
}

function __webgl_enable_WEBGL_draw_instanced_base_vertex_base_instance(ctx) {
    // Closure is expected to be allowed to minify the '.dibvbi' property, so not
accessing it quoted.
    return !(ctx.dibvbi =
ctx.getExtension('WEBGL_draw_instanced_base_vertex_base_instance'));
}

function __webgl_enable_WEBGL_multi_draw_instanced_base_vertex_base_instance(ctx) {
    // Closure is expected to be allowed to minify the '.mdibvbi' property, so not
accessing it quoted.
    return !(ctx.mdibvbi =
ctx.getExtension('WEBGL_multi_draw_instanced_base_vertex_base_instance'));
}

function __webgl_enable_WEBGL_multi_draw(ctx) {
    // Closure is expected to be allowed to minify the '.multiDrawWebgl' property, so
not accessing it quoted.
    return !(ctx.multiDrawWebgl = ctx.getExtension('WEBGL_multi_draw'));
}
var GL =
{counter:1, buffers:[], mappedBuffers:{}, programs:[], framebuffers:[], renderbuffers:[], tex
tures:[], shaders:[], vaos:[], contexts:[], offscreenCanvases:{}, queries:[], samplers:[], tra
nsformFeedbacks:[], syncs:[], byteSizeByTypeRoot:5120, byteSizeByType:[1,1,2,2,4,4,4,2,3,4
,8], stringCache:{}, stringiCache:{}, unpackAlignment:4, recordError:function
recordError(errorCode) {
    if (!GL.lastError) {
        GL.lastError = errorCode;
    }
}, getNewId:function(table) {
    var ret = GL.counter++;
    for (var i = table.length; i < ret; i++) {
        table[i] = null;
    }
}
}

```

```

    return ret;

},MAX_TEMP_BUFFER_SIZE:2097152,numTempVertexBuffersPerSize:64,log2ceilLookup:function(i)
) {
    return 32 - Math.clz32(i === 0 ? 0 : i - 1);
},generateTempBuffers:function(quads, context) {
    var largestIndex = GL.log2ceilLookup(GL.MAX_TEMP_BUFFER_SIZE);
    context.tempVertexBufferCounters1 = [];
    context.tempVertexBufferCounters2 = [];
    context.tempVertexBufferCounters1.length =
context.tempVertexBufferCounters2.length = largestIndex+1;
    context.tempVertexBuffers1 = [];
    context.tempVertexBuffers2 = [];
    context.tempVertexBuffers1.length = context.tempVertexBuffers2.length =
largestIndex+1;
    context.tempIndexBuffers = [];
    context.tempIndexBuffers.length = largestIndex+1;
    for (var i = 0; i <= largestIndex; ++i) {
        context.tempIndexBuffers[i] = null; // Created on-demand
        context.tempVertexBufferCounters1[i] = context.tempVertexBufferCounters2[i] =
0;
        var ringbufferLength = GL.numTempVertexBuffersPerSize;
        context.tempVertexBuffers1[i] = [];
        context.tempVertexBuffers2[i] = [];
        var ringbuffer1 = context.tempVertexBuffers1[i];
        var ringbuffer2 = context.tempVertexBuffers2[i];
        ringbuffer1.length = ringbuffer2.length = ringbufferLength;
        for (var j = 0; j < ringbufferLength; ++j) {
            ringbuffer1[j] = ringbuffer2[j] = null; // Created on-demand
        }
    }

    if (quads) {
        // GL_QUAD indexes can be precalculated
        context.tempQuadIndexBuffer = GLctx.createBuffer();
        context.GLctx.bindBuffer(0x8893 /*GL_ELEMENT_ARRAY_BUFFER*/,
context.tempQuadIndexBuffer);
        var numIndexes = GL.MAX_TEMP_BUFFER_SIZE >> 1;
        var quadIndexes = new Uint16Array(numIndexes);
        var i = 0, v = 0;
        while (1) {
            quadIndexes[i++] = v;
            if (i >= numIndexes) break;
            quadIndexes[i++] = v+1;
            if (i >= numIndexes) break;
            quadIndexes[i++] = v+2;
            if (i >= numIndexes) break;
            quadIndexes[i++] = v;
            if (i >= numIndexes) break;
            quadIndexes[i++] = v+2;
            if (i >= numIndexes) break;
            quadIndexes[i++] = v+3;
            if (i >= numIndexes) break;
            v += 4;
        }
    }
}

```

```

        context.GLctx.bufferData(0x8893 /*GL_ELEMENT_ARRAY_BUFFER*/, quadIndexes,
0x88E4 /*GL_STATIC_DRAW*/);
        context.GLctx.bindBuffer(0x8893 /*GL_ELEMENT_ARRAY_BUFFER*/, null);
    }
},getTempVertexBuffer:function getTempVertexBuffer(sizeBytes) {
    var idx = GL.log2ceilLookup(sizeBytes);
    var ringbuffer = GL.currentContext.tempVertexBuffers1[idx];
    var nextFreeBufferIndex = GL.currentContext.tempVertexBufferCounters1[idx];
    GL.currentContext.tempVertexBufferCounters1[idx] =
(GL.currentContext.tempVertexBufferCounters1[idx]+1) & (GL.numTempVertexBuffersPerSize-
1);
    var vbo = ringbuffer[nextFreeBufferIndex];
    if (vbo) {
        return vbo;
    }
    var prevVBO = GLctx.getParameter(0x8894 /*GL_ARRAY_BUFFER_BINDING*/);
    ringbuffer[nextFreeBufferIndex] = GLctx.createBuffer();
    GLctx.bindBuffer(0x8892 /*GL_ARRAY_BUFFER*/, ringbuffer[nextFreeBufferIndex]);
    GLctx.bufferData(0x8892 /*GL_ARRAY_BUFFER*/, 1 << idx, 0x88E8
/*GL_DYNAMIC_DRAW*/);
    GLctx.bindBuffer(0x8892 /*GL_ARRAY_BUFFER*/, prevVBO);
    return ringbuffer[nextFreeBufferIndex];
},getTempIndexBuffer:function getTempIndexBuffer(sizeBytes) {
    var idx = GL.log2ceilLookup(sizeBytes);
    var ibo = GL.currentContext.tempIndexBuffers[idx];
    if (ibo) {
        return ibo;
    }
    var prevIBO = GLctx.getParameter(0x8895 /*ELEMENT_ARRAY_BUFFER_BINDING*/);
    GL.currentContext.tempIndexBuffers[idx] = GLctx.createBuffer();
    GLctx.bindBuffer(0x8893 /*GL_ELEMENT_ARRAY_BUFFER*/,
GL.currentContext.tempIndexBuffers[idx]);
    GLctx.bufferData(0x8893 /*GL_ELEMENT_ARRAY_BUFFER*/, 1 << idx, 0x88E8
/*GL_DYNAMIC_DRAW*/);
    GLctx.bindBuffer(0x8893 /*GL_ELEMENT_ARRAY_BUFFER*/, prevIBO);
    return GL.currentContext.tempIndexBuffers[idx];
},newRenderingFrameStarted:function newRenderingFrameStarted() {
    if (!GL.currentContext) {
        return;
    }
    var vb = GL.currentContext.tempVertexBuffers1;
    GL.currentContext.tempVertexBuffers1 = GL.currentContext.tempVertexBuffers2;
    GL.currentContext.tempVertexBuffers2 = vb;
    vb = GL.currentContext.tempVertexBufferCounters1;
    GL.currentContext.tempVertexBufferCounters1 =
GL.currentContext.tempVertexBufferCounters2;
    GL.currentContext.tempVertexBufferCounters2 = vb;
    var largestIndex = GL.log2ceilLookup(GL.MAX_TEMP_BUFFER_SIZE);
    for (var i = 0; i <= largestIndex; ++i) {
        GL.currentContext.tempVertexBufferCounters1[i] = 0;
    }
},getSource:function(shader, count, string, length) {
    var source = '';
    for (var i = 0; i < count; ++i) {
        var len = length ? HEAP32[((length)+(i*4))>>2] : -1;
        source += UTF8ToString(HEAP32[((string)+(i*4))>>2]), len < 0 ? undefined :
len);
}

```

```

    }
    return source;
},calcBufLength:function calcBufLength(size, type, stride, count) {
    if (stride > 0) {
        return count * stride; // XXXvlad this is not exactly correct I don't think
    }
    var typeSize = GL.byteSizeByType[type - GL.byteSizeByTypeRoot];
    return size * typeSize * count;
},usedTempBuffers:[],preDrawHandleClientVertexAttribBindings:function
preDrawHandleClientVertexAttribBindings(count) {
    GL.resetBufferBinding = false;

    // TODO: initial pass to detect ranges we need to upload, might not need an
    upload per attrib
    for (var i = 0; i < GL.currentContext.maxVertexAttribs; ++i) {
        var cb = GL.currentContext.clientBuffers[i];
        if (!cb.clientside || !cb.enabled) continue;

        GL.resetBufferBinding = true;

        var size = GL.calcBufLength(cb.size, cb.type, cb.stride, count);
        var buf = GL.getTempVertexBuffer(size);
        GLctx.bindBuffer(0x8892 /*GL_ARRAY_BUFFER*/, buf);
        GLctx.bufferSubData(0x8892 /*GL_ARRAY_BUFFER*/,
            0,
            HEAPU8.subarray(cb.ptr, cb.ptr + size));
        cb.vertexAttribPointerAdaptor.call(GLctx, i, cb.size, cb.type, cb.normalized,
        cb.stride, 0);
    }
},postDrawHandleClientVertexAttribBindings:function
postDrawHandleClientVertexAttribBindings() {
    if (GL.resetBufferBinding) {
        GLctx.bindBuffer(0x8892 /*GL_ARRAY_BUFFER*/,
        GL.buffers[GLctx.currentArrayBufferBinding]);
    }
},createContext:function/** @type {HTMLCanvasElement} */ canvas,
webGLContextAttributes) {

    // BUG: Workaround Safari WebGL issue: After successfully acquiring WebGL
    context on a canvas,
    // calling .getContext() will always return that context independent of which
    'webgl' or 'webgl2'
    // context version was passed. See
    https://bugs.webkit.org/show_bug.cgi?id=222758 and
    // https://github.com/emscripten-core/emscripten/issues/13295.
    // TODO: Once the bug is fixed and shipped in Safari, adjust the Safari version
    field in above check.
    if (!canvas.getContextSafariWebGL2Fixed) {
        canvas.getContextSafariWebGL2Fixed = canvas.getContext;
        /** @type {function(this:HTMLCanvasElement, string, (Object|null)=):
        (Object|null)} */
        function fixedGetContext(ver, attrs) {
            var gl = canvas.getContextSafariWebGL2Fixed(ver, attrs);
            return ((ver == 'webgl') == (gl instanceof WebGLRenderingContext)) ? gl :
        null;
    }
}

```

```

        }
        canvas.getContext = fixedGetContext;
    }

    var ctx =
        (webGLContextAttributes.majorVersion > 1)
    ?
        canvas.getContext("webgl2", webGLContextAttributes)
    :
        (canvas.getContext("webgl", webGLContextAttributes)
        // https://caniuse.com/#feat=webgl
        );
}

if (!ctx) return 0;

var handle = GL.registerContext(ctx, webGLContextAttributes);

return handle;
},registerContext:function(ctx, webGLContextAttributes) {
// without pthreads a context is just an integer ID
var handle = GL.getNewId(GL.contexts);

var context = {
    handle: handle,
    attributes: webGLContextAttributes,
    version: webGLContextAttributes.majorVersion,
    GLctx: ctx
};

// Store the created context object so that we can access the context given a
// canvas without having to pass the parameters again.
if (ctx.canvas) ctx.canvas.GLctxObject = context;
GL.contexts[handle] = context;
if (typeof webGLContextAttributes.enableExtensionsByDefault == 'undefined' ||
webGLContextAttributes.enableExtensionsByDefault) {
    GL.initExtensions(context);
}

context.maxVertexAttribs = context.GLctx.getParameter(0x8869
/*GL_MAX_VERTEX_ATTRIBS*/);
context.clientBuffers = [];
for (var i = 0; i < context.maxVertexAttribs; i++) {
    context.clientBuffers[i] = { enabled: false, clientside: false, size: 0,
type: 0, normalized: 0, stride: 0, ptr: 0, vertexAttribPointerAdaptor: null };
}

GL.generateTempBuffers(false, context);

return handle;
},makeContextCurrent:function(contextHandle) {

    GL.currentContext = GL.contexts[contextHandle]; // Active Emscripten GL layer
    context object.
    Module.ctx = GLctx = GL.currentContext && GL.currentContext.GLctx; // Active
    WebGL context object.
}

```

```

    return !(contextHandle && !GLctx);
},getContext:function(contextHandle) {
    return GL.contexts[contextHandle];
},deleteContext:function(contextHandle) {
    if (GL.currentContext === GL.contexts[contextHandle]) GL.currentContext = null;
    if (typeof JSEvents == 'object')
JSEvents.removeAllHandlersOnTarget(GL.contexts[contextHandle].GLctx.canvas); // Release
all JS event handlers on the DOM element that the GL context is associated with since
the context is now deleted.
    if (GL.contexts[contextHandle] && GL.contexts[contextHandle].GLctx.canvas)
GL.contexts[contextHandle].GLctx.canvas.GLctxObject = undefined; // Make sure the
canvas object no longer refers to the context object so there are no GC surprises.
    GL.contexts[contextHandle] = null;
},initExtensions:function(context) {
    // If this function is called without a specific context object, init the
extensions of the currently active context.
    if (!context) context = GL.currentContext;

    if (context.initExtensionsDone) return;
    context.initExtensionsDone = true;

    var GLctx = context.GLctx;

    // Detect the presence of a few extensions manually, this GL interop layer
itself will need to know if they exist.

    // Extensions that are only available in WebGL 1 (the calls will be no-ops if
called on a WebGL 2 context active)
    __webgl_enable_ANGLE_instanced_arrays(GLctx);
    __webgl_enable_OES_vertex_array_object(GLctx);
    __webgl_enable_WEBGL_draw_buffers(GLctx);
    // Extensions that are available from WebGL >= 2 (no-op if called on a WebGL 1
context active)
    __webgl_enable_WEBGL_draw_instanced_base_vertex_base_instance(GLctx);
    __webgl_enable_WEBGL_multi_draw_instanced_base_vertex_base_instance(GLctx);

    // On WebGL 2, EXT_disjoint_timer_query is replaced with an alternative
    // that's based on core APIs, and exposes only the queryCounterEXT()
    // entrypoint.
    if (context.version >= 2) {
        GLctx.disjointTimerQueryExt =
GLctx.getExtension("EXT_disjoint_timer_query_webgl2");
    }

    // However, Firefox exposes the WebGL 1 version on WebGL 2 as well and
    // thus we look for the WebGL 1 version again if the WebGL 2 version
    // isn't present. https://bugzilla.mozilla.org/show_bug.cgi?id=1328882
    if (context.version < 2 || !GLctx.disjointTimerQueryExt)
    {
        GLctx.disjointTimerQueryExt = GLctx.getExtension("EXT_disjoint_timer_query");
    }

    __webgl_enable_WEBGL_multi_draw(GLctx);

    // .getSupportedExtensions() can return null if context is lost, so coerce to
empty array.

```

```

var exts = GLctx.getSupportedExtensions() || [];
exts.forEach(function(ext) {
    // WEBGL_lose_context, WEBGL_debug_renderer_info and WEBGL_debug_shaders are
not enabled by default.
    if (!ext.includes('lose_context') && !ext.includes('debug')) {
        // Call .getExtension() to enable that extension permanently.
        GLctx.getExtension(ext);
    }
});
}};

var __emscripten_webgl_power_preferences = ['default', 'low-power', 'high-
performance'];
function _emscripten_webgl_do_create_context(target, attributes) {
    assert(attributes);
    var a = attributes >> 2;
    var powerPreference = HEAP32[a + (24>>2)];
    var contextAttributes = {
        'alpha': !!HEAP32[a + (0>>2)],
        'depth': !!HEAP32[a + (4>>2)],
        'stencil': !!HEAP32[a + (8>>2)],
        'antialias': !!HEAP32[a + (12>>2)],
        'premultipliedAlpha': !!HEAP32[a + (16>>2)],
        'preserveDrawingBuffer': !!HEAP32[a + (20>>2)],
        'powerPreference': __emscripten_webgl_power_preferences[powerPreference],
        'failIfMajorPerformanceCaveat': !!HEAP32[a + (28>>2)],
        // The following are not predefined WebGL context attributes in the WebGL
specification, so the property names can be minified by Closure.
        majorVersion: HEAP32[a + (32>>2)],
        minorVersion: HEAP32[a + (36>>2)],
        enableExtensionsByDefault: HEAP32[a + (40>>2)],
        explicitSwapControl: HEAP32[a + (44>>2)],
        proxyContextToMainThread: HEAP32[a + (48>>2)],
        renderViaOffscreenBackBuffer: HEAP32[a + (52>>2)]
    };
}

var canvas = findCanvasEventTarget(target);

if (!canvas) {
    return 0;
}

if (contextAttributes.explicitSwapControl) {
    return 0;
}

var contextHandle = GL.createContext(canvas, contextAttributes);
return contextHandle;
}
function _emscripten_webgl_create_context(a0,a1
) {
return _emscripten_webgl_do_create_context(a0,a1);
}

function _emscripten_webgl_destroy_context(contextHandle) {

```

```

    if (GL.currentContext == contextHandle) GL.currentContext = 0;
    GL.deleteContext(contextHandle);
}

function _emscripten_webgl_enable_extension(contextHandle, extension) {
    var context = GL.getContext(contextHandle);
    var extString = UTF8ToString(extension);
    if (extString.startsWith('GL_')) extString = extString.substr(3); // Allow
enabling extensions both with "GL_" prefix and without.

    // Switch-board that pulls in code for all GL extensions, even if those are not
used :/
    // Build with -s GL_SUPPORT_SIMPLE_ENABLE_EXTENSIONS = 0 to avoid this.

    // Obtain function entry points to WebGL 1 extension related functions.
    if (extString == 'ANGLE_instanced_arrays')
__webgl_enable_ANGLE_instanced_arrays(GLctx);
    if (extString == 'OES_vertex_array_object')
__webgl_enable_OES_vertex_array_object(GLctx);
    if (extString == 'WEBGL_draw_buffers') __webgl_enable_WEBGL_draw_buffers(GLctx);

    if (extString == 'WEBGL_draw_instanced_base_vertex_base_instance')
__webgl_enable_WEBGL_draw_instanced_base_vertex_base_instance(GLctx);
    if (extString == 'WEBGL_multi_draw_instanced_base_vertex_base_instance')
__webgl_enable_WEBGL_multi_draw_instanced_base_vertex_base_instance(GLctx);

    if (extString == 'WEBGL_multi_draw') __webgl_enable_WEBGL_multi_draw(GLctx);

    var ext = context.GLctx.getExtension(extString);
    return !ext;
}

function _emscripten_webgl_do_get_current_context() {
    return GL.currentContext ? GL.currentContext.handle : 0;
}
function _emscripten_webgl_get_current_context(
) {
return _emscripten_webgl_do_get_current_context();
}

function _emscripten_webgl_init_context_attributes(attributes) {
    assert(attributes);
    var a = attributes >> 2;
    for (var i = 0; i < (56>>2); ++i) {
        HEAP32[a+i] = 0;
    }

    HEAP32[a + (0>>2)] =
    HEAP32[a + (4>>2)] =
    HEAP32[a + (12>>2)] =
    HEAP32[a + (16>>2)] =
    HEAP32[a + (32>>2)] =
    HEAP32[a + (40>>2)] = 1;
}

```

```

function _emscripten_webgl_make_context_current(contextHandle) {
    var success = GL.makeContextCurrent(contextHandle);
    return success ? 0 : -5;
}

var ENV = {};

function getExecutableName() {
    return thisProgram || './this.program';
}

function getEnvStrings() {
    if (!getEnvStrings.strings) {
        // Default values.
        // Browser language detection #8751
        var lang = ((typeof navigator == 'object' && navigator.languages &&
navigator.languages[0]) || 'C').replace('-', '_') + '.UTF-8';
        var env = {
            'USER': 'web_user',
            'LOGNAME': 'web_user',
            'PATH': '/',
            'PWD': '/',
            'HOME': '/home/web_user',
            'LANG': lang,
            '_': getExecutableName()
        };
        // Apply the user-provided values, if any.
        for (var x in ENV) {
            // x is a key in ENV; if ENV[x] is undefined, that means it was
            // explicitly set to be so. We allow user code to do that to
            // force variables with default values to remain unset.
            if (ENV[x] === undefined) delete env[x];
            else env[x] = ENV[x];
        }
        var strings = [];
        for (var x in env) {
            strings.push(x + '=' + env[x]);
        }
        getEnvStrings.strings = strings;
    }
    return getEnvStrings.strings;
}

function _environ_get(__environ, environ_buf) {
    var bufSize = 0;
    getEnvStrings().forEach(function(string, i) {
        var ptr = environ_buf + bufSize;
        HEAP32[((__environ)+(i * 4))>>2] = ptr;
        writeAsciiToMemory(string, ptr);
        bufSize += string.length + 1;
    });
    return 0;
}

function _environ_sizes_get(penviron_count, penviron_buf_size) {
    var strings = getEnvStrings();

```

```

    HEAP32[((pviron_count)>>2)] = strings.length;
    var bufSize = 0;
    strings.forEach(function(string) {
        bufSize += string.length + 1;
    });
    HEAP32[((pviron_buf_size)>>2)] = bufSize;
    return 0;
}

function _fd_close(fd) {
try {

    var stream = SYSCALLS.getStreamFromFD(fd);
    FS.close(stream);
    return 0;
} catch (e) {
    if (typeof FS == 'undefined' || !(e instanceof FS.ErrnoError)) throw e;
    return e(errno);
}
}

function _fd_fdstat_get(fd, pbuf) {
try {

    var stream = SYSCALLS.getStreamFromFD(fd);
    // All character devices are terminals (other things a Linux system would
    // assume is a character device, like the mouse, we have special APIs for).
    var type = stream.tty ? 2 :
               FS.isDir(stream.mode) ? 3 :
               FS.isLink(stream.mode) ? 7 :
               4;
    HEAP8[((pbuf)>>0)] = type;
    // TODO HEAP16[((pbuf)+(2))>>1] = ?;
    // TODO (tempI64 = [?>>0, (tempDouble=?, (+(Math.abs(tempDouble))) >= 1.0 ?
    (tempDouble > 0.0 ? ((Math.min((+(Math.floor((tempDouble)/4294967296.0))), 4294967295.0))|0)>>0 : (~(~((+(Math.ceil((tempDouble -
+((~(tempDouble))>>0))/4294967296.0))))>>0) : 0)], HEAP32[((pbuf)+(8))>>2] =
    tempI64[0], HEAP32[((pbuf)+(12))>>2] = tempI64[1];
    // TODO (tempI64 = [?>>0, (tempDouble=?, (+(Math.abs(tempDouble))) >= 1.0 ?
    (tempDouble > 0.0 ? ((Math.min((+(Math.floor((tempDouble)/4294967296.0))), 4294967295.0))|0)>>0 : (~(~((+(Math.ceil((tempDouble -
+((~(tempDouble))>>0))/4294967296.0))))>>0) : 0)], HEAP32[((pbuf)+(16))>>2] =
    tempI64[0], HEAP32[((pbuf)+(20))>>2] = tempI64[1]);
    return 0;
} catch (e) {
    if (typeof FS == 'undefined' || !(e instanceof FS.ErrnoError)) throw e;
    return e(errno);
}
}

function _fd_read(fd, iov, iovcnt, pnum) {
try {

    var stream = SYSCALLS.getStreamFromFD(fd);

```

```

    var num = SYSCALLS.doReadv(stream, iov, iovcnt);
    HEAP32[((pnum)>>2)] = num;
    return 0;
} catch (e) {
    if (typeof FS == 'undefined' || !(e instanceof FS.ErrnoError)) throw e;
    return e(errno);
}

function _fd_seek(fd, offset_low, offset_high, whence, newOffset) {
try {

    var stream = SYSCALLS.getStreamFromFD(fd);
    var HIGH_OFFSET = 0x100000000; // 2^32
    // use an unsigned operator on low and shift high by 32-bits
    var offset = offset_high * HIGH_OFFSET + (offset_low >>> 0);

    var DOUBLE_LIMIT = 0x2000000000000000; // 2^53
    // we also check for equality since DOUBLE_LIMIT + 1 == DOUBLE_LIMIT
    if (offset <= -DOUBLE_LIMIT || offset >= DOUBLE_LIMIT) {
        return -61;
    }

    FS.llseek(stream, offset, whence);
    (tempI64 =
[stream.position>>>0, (tempDouble=stream.position, (+Math.abs(tempDouble))) >= 1.0 ?
(tempDouble > 0.0 ? ((Math.min((+(Math.floor((tempDouble)/4294967296.0))), 4294967295.0))|0)>>>0 : (~((+(Math.ceil((tempDouble -
+((~(tempDouble))>>>0))/4294967296.0))))>>>0) : 0)],HEAP32[((newOffset)>>2)] =
tempI64[0],HEAP32[((newOffset)+(4))>>2]] = tempI64[1]);
        if (stream.getdents && offset === 0 && whence === 0) stream.getdents = null; //
reset readdir state
        return 0;
    } catch (e) {
        if (typeof FS == 'undefined' || !(e instanceof FS.ErrnoError)) throw e;
        return e(errno);
    }
}

function _fd_write(fd, iov, iovcnt, pnum) {
try {

    ;
    var stream = SYSCALLS.getStreamFromFD(fd);
    var num = SYSCALLS.doWritev(stream, iov, iovcnt);
    HEAP32[((pnum)>>2)] = num;
    return 0;
} catch (e) {
    if (typeof FS == 'undefined' || !(e instanceof FS.ErrnoError)) throw e;
    return e(errno);
}

function _getTempRet0() {
    return getTempRet0();
}

```

```

}

function getHostByName(name) {
    // generate hostent
    var ret = _malloc(20); // XXX possibly leaked, as are others here
    var nameBuf = _malloc(name.length+1);
    stringToUTF8(name, nameBuf, name.length+1);
    HEAP32[((ret)>>2)] = nameBuf;
    var aliasesBuf = _malloc(4);
    HEAP32[((aliasesBuf)>>2)] = 0;
    HEAP32[((((ret)+(4))>>2))] = aliasesBuf;
    var afinet = 2;
    HEAP32[((((ret)+(8))>>2))] = afinet;
    HEAP32[((((ret)+(12))>>2))] = 4;
    var addrListBuf = _malloc(12);
    HEAP32[((addrListBuf)>>2)] = addrListBuf+8;
    HEAP32[((((addrListBuf)+(4))>>2))] = 0;
    HEAP32[((((addrListBuf)+(8))>>2))] = inetPton4(DNS.lookup_name(name));
    HEAP32[((((ret)+(16))>>2))] = addrListBuf;
    return ret;
}
function _gethostbyaddr(addr, addrlen, type) {
    if (type !== 2) {
        setErrNo(5);
        // TODO: set h_errno
        return null;
    }
    addr = HEAP32[((addr)>>2)]; // addr is in_addr
    var host = inetNtop4(addr);
    var lookup = DNS.lookup_addr(host);
    if (lookup) {
        host = lookup;
    }
    return getHostByName(host);
}

function _gethostbyname(name) {
    return getHostByName(UTF8ToString(name));
}

function _glActiveTexture(x0) { GLctx['activeTexture'](x0) }

function _glAttachShader(program, shader) {
    program = GL.programs[program];
    shader = GL.shaders[shader];
    program[shader.shaderType] = shader;
    GLctx.attachShader(program, shader);
}

function _glBeginQuery(target, id) {
    GLctx['beginQuery'](target, GL.queries[id]);
}

function _glBindAttribLocation(program, index, name) {
    GLctx.bindAttribLocation(GL.programs[program], index, UTF8ToString(name));
}

```

```

}

function _glBindBuffer(target, buffer) {
    if (target == 0x8892 /*GL_ARRAY_BUFFER*/) {
        GLctx.currentArrayBufferBinding = buffer;
    } else if (target == 0x8893 /*GL_ELEMENT_ARRAY_BUFFER*/) {
        GLctx.currentElementArrayBufferBinding = buffer;
    }

    if (target == 0x88EB /*GL_PIXEL_PACK_BUFFER*/) {
        // In WebGL 2 glReadPixels entry point, we need to use a different WebGL 2 API
        // function call when a buffer is bound to
        // GL_PIXEL_PACK_BUFFER_BINDING point, so must keep track whether that binding
        // point is non-null to know what is
        // the proper API function to call.
        GLctx.currentPixelPackBufferBinding = buffer;
    } else if (target == 0x88EC /*GL_PIXEL_UNPACK_BUFFER*/) {
        // In WebGL 2 gl(Compressed)Tex(Sub)Image[23]D entry points, we need to
        // use a different WebGL 2 API function call when a buffer is bound to
        // GL_PIXEL_UNPACK_BUFFER_BINDING point, so must keep track whether that
        // binding point is non-null to know what is the proper API function to
        // call.
        GLctx.currentPixelUnpackBufferBinding = buffer;
    }
    GLctx.bindBuffer(target, GL.buffers[buffer]);
}

function _glBindBufferBase(target, index, buffer) {
    GLctx['bindBufferBase'](target, index, GL.buffers[buffer]);
}

function _glBindBufferRange(target, index, buffer, offset, ptrsize) {
    GLctx['bindBufferRange'](target, index, GL.buffers[buffer], offset, ptrsize);
}

function _glBindFramebuffer(target, framebuffer) {
    GLctx.bindFramebuffer(target, GL.framebuffers[framebuffer]);
}

function _glBindRenderbuffer(target, renderbuffer) {
    GLctx.bindRenderbuffer(target, GL.renderbuffers[renderbuffer]);
}

function _glBindSampler(unit, sampler) {
    GLctx['bindSampler'](unit, GL.samplers[sampler]);
}

function _glBindTexture(target, texture) {
    GLctx.bindTexture(target, GL.textures[texture]);
}

function _glBindVertexArray(vao) {
    GLctx['bindVertexArray'](GL.vaos[vao]);
}

```

```

    var ibo = GLctx.getParameter(0x8895 /*ELEMENT_ARRAY_BUFFER_BINDING*/);
    GLctx.currentElementArrayBufferBinding = ibo ? (ibo.name | 0) : 0;
}

function _glBlendEquation(x0) { GLctx['blendEquation'](x0) }

function _glBlendEquationSeparate(x0, x1) { GLctx['blendEquationSeparate'](x0, x1) }

function _glBlendFuncSeparate(x0, x1, x2, x3) { GLctx['blendFuncSeparate'](x0, x1,
x2, x3) }

function _glBlitFramebuffer(x0, x1, x2, x3, x4, x5, x6, x7, x8, x9) {
GLctx['blitFramebuffer'](x0, x1, x2, x3, x4, x5, x6, x7, x8, x9) }

function _glBufferData(target, size, data, usage) {

    if (GL.currentContext.version >= 2) { // WebGL 2 provides new garbage-free entry
points to call to WebGL. Use those always when possible.
        if (data) {
            GLctx.bufferData(target, HEAPU8, usage, data, size);
        } else {
            GLctx.bufferData(target, size, usage);
        }
    } else {
        // N.b. here first form specifies a heap subarray, second form an integer size,
so the ?: code here is polymorphic. It is advised to avoid
        // randomly mixing both uses in calling code, to avoid any potential JS engine
JIT issues.
        GLctx.bufferData(target, data ? HEAPU8.subarray(data, data+size) : size,
usage);
    }
}

function _glBufferSubData(target, offset, size, data) {
    if (GL.currentContext.version >= 2) { // WebGL 2 provides new garbage-free entry
points to call to WebGL. Use those always when possible.
        GLctx.bufferSubData(target, offset, HEAPU8, data, size);
        return;
    }
    GLctx.bufferSubData(target, offset, HEAPU8.subarray(data, data+size));
}

function _glCheckFramebufferStatus(x0) { return GLctx['checkFramebufferStatus'](x0) }

function _glClear(x0) { GLctx['clear'](x0) }

function _glClearBufferfi(x0, x1, x2, x3) { GLctx['clearBufferfi'](x0, x1, x2, x3) }

function _glClearBufferfv(buffer, drawbuffer, value) {

    GLctx['clearBufferfv'](buffer, drawbuffer, HEAPF32, value>>2);
}

function _glClearBufferuiv(buffer, drawbuffer, value) {

```

```

    GLctx['clearBufferuiv'](buffer, drawbuffer, HEAPU32, value>>2);
}

function _glClearColor(x0, x1, x2, x3) { GLctx['clearColor'](x0, x1, x2, x3) }

function _glClearDepthf(x0) { GLctx['clearDepth'](x0) }

function _glClearStencil(x0) { GLctx['clearStencil'](x0) }

function _glClientWaitSync(sync, flags, timeoutLo, timeoutHi) {
    // WebGL2 vs GLES3 differences: in GLES3, the timeout parameter is a uint64,
    where 0xFFFFFFFFFFFFFFFULL means GL_TIMEOUT_IGNORED.
    // In JS, there's no 64-bit value types, so instead timeout is taken to be
    signed, and GL_TIMEOUT_IGNORED is given value -1.
    // Inherently the value accepted in the timeout is lossy, and can't take in
    arbitrary u64 bit pattern (but most likely doesn't matter)
    // See https://www.khronos.org/registry/webgl/specs/latest/2.0/#5.15
    return GLctx.clientWaitSync(GL.syncs[sync], flags, convertI32PairToI53(timeoutLo,
    timeoutHi));
}

function _glColorMask(red, green, blue, alpha) {
    GLctx.colorMask(!red, !green, !blue, !alpha);
}

function _glCompileShader(shader) {
    GLctx.compileShader(GL.shaders[shader]);
}

function _glCompressedTexImage2D(target, level, internalFormat, width, height,
border, imageSize, data) {
    if (GL.currentContext.version >= 2) { // WebGL 2 provides new garbage-free entry
    points to call to WebGL. Use those always when possible.
        if (GLctx.currentPixelUnpackBufferBinding) {
            GLctx['compressedTexImage2D'](target, level, internalFormat, width, height,
            border, imageSize, data);
        } else {
            GLctx['compressedTexImage2D'](target, level, internalFormat, width, height,
            border, HEAPU8, data, imageSize);
        }
        return;
    }
    GLctx['compressedTexImage2D'](target, level, internalFormat, width, height,
    border, data ? HEAPU8.subarray((data), (data+imageSize)) : null);
}

function _glCompressedTexImage3D(target, level, internalFormat, width, height, depth,
border, imageSize, data) {
    if (GLctx.currentPixelUnpackBufferBinding) {
        GLctx['compressedTexImage3D'](target, level, internalFormat, width, height,
        depth, border, imageSize, data);
    } else {
        GLctx['compressedTexImage3D'](target, level, internalFormat, width, height,
        depth, border, HEAPU8, data, imageSize);
    }
}

```

```

}

function _glCompressedTexSubImage2D(target, level, xoffset, yoffset, width, height,
format, imageSize, data) {
    if (GL.currentContext.version >= 2) { // WebGL 2 provides new garbage-free entry
points to call to WebGL. Use those always when possible.
        if (GLctx.currentPixelUnpackBufferBinding) {
            GLctx['compressedTexSubImage2D'](target, level, xoffset, yoffset, width,
height, format, imageSize, data);
        } else {
            GLctx['compressedTexSubImage2D'](target, level, xoffset, yoffset, width,
height, format, HEAPU8, data, imageSize);
        }
        return;
    }
    GLctx['compressedTexSubImage2D'](target, level, xoffset, yoffset, width, height,
format, data ? HEAPU8.subarray((data), (data+imageSize)) : null);
}

function _glCompressedTexSubImage3D(target, level, xoffset, yoffset, zoffset, width,
height, depth, format, imageSize, data) {
    if (GLctx.currentPixelUnpackBufferBinding) {
        GLctx['compressedTexSubImage3D'](target, level, xoffset, yoffset, zoffset,
width, height, depth, format, imageSize, data);
    } else {
        GLctx['compressedTexSubImage3D'](target, level, xoffset, yoffset, zoffset,
width, height, depth, format, HEAPU8, data, imageSize);
    }
}

function _glCopyBufferSubData(x0, x1, x2, x3, x4) { GLctx['copyBufferSubData'](x0,
x1, x2, x3, x4) }

function _glCopyTexImage2D(x0, x1, x2, x3, x4, x5, x6, x7) { GLctx['copyTexImage2D'](x0,
x1, x2, x3, x4, x5, x6, x7) }

function _glCopyTexSubImage2D(x0, x1, x2, x3, x4, x5, x6, x7) { GLctx['copyTexSubImage2D'](x0,
x1, x2, x3, x4, x5, x6, x7) }

function _glCreateProgram() {
    var id = GL.getId(GL.programs);
    var program = GLctx.createProgram();
    // Store additional information needed for each shader program:
    program.name = id;
    // Lazy cache results of
    glGetProgramiv(GL_ACTIVE_UNIFORM_MAX_LENGTH/GL_ACTIVE_ATTRIBUTE_MAX_LENGTH/GL_ACTIVE_UN
IFORM_BLOCK_MAX_NAME_LENGTH)
    program.maxUniformLength = program.maxAttributeLength =
program.maxUniformBlockNameLength = 0;
    program.uniformIdCounter = 1;
    GL.programs[id] = program;
    return id;
}

function _glCreateShader(shaderType) {
    var id = GL.getId(GL.shaders);
}

```

```

GL.shaders[id] = GLctx.createShader(shaderType);

// GL_VERTEX_SHADER = 0x8B31, GL_FRAGMENT_SHADER = 0x8B30
GL.shaders[id].shaderType = shaderType&1?'vs':'fs';

return id;
}

function _glCullFace(x0) { GLctx['cullFace'](x0) }

function _glDeleteBuffers(n, buffers) {
for (var i = 0; i < n; i++) {
var id = HEAP32[((buffers)+(i*4))>>2];
var buffer = GL.buffers[id];

// From spec: "glDeleteBuffers silently ignores 0's and names that do not
// correspond to existing buffer objects."
if (!buffer) continue;

GLctx.deleteBuffer(buffer);
buffer.name = 0;
GL.buffers[id] = null;

if (id == GLctx.currentArrayBufferBinding) GLctx.currentArrayBufferBinding = 0;
if (id == GLctx.currentElementArrayBufferBinding)
GLctx.currentElementArrayBufferBinding = 0;
if (id == GLctx.currentPixelPackBufferBinding)
GLctx.currentPixelPackBufferBinding = 0;
if (id == GLctx.currentPixelUnpackBufferBinding)
GLctx.currentPixelUnpackBufferBinding = 0;
}
}

function _glDeleteFramebuffers(n, framebuffers) {
for (var i = 0; i < n; ++i) {
var id = HEAP32[((framebuffers)+(i*4))>>2];
var framebuffer = GL.framebuffers[id];
if (!framebuffer) continue; // GL spec: "glDeleteFramebuffers silently ignores
0s and names that do not correspond to existing framebuffer objects".
GLctx.deleteFramebuffer(framebuffer);
framebuffer.name = 0;
GL.framebuffers[id] = null;
}
}

function _glDeleteProgram(id) {
if (!id) return;
var program = GL.programs[id];
if (!program) { // glDeleteProgram actually signals an error when deleting a
nonexisting object, unlike some other GL delete functions.
GL.recordError(0x501 /* GL_INVALID_VALUE */);
return;
}
GLctx.deleteProgram(program);
program.name = 0;
}

```

```

    GL.programs[id] = null;
}

function _glDeleteQueries(n, ids) {
    for (var i = 0; i < n; i++) {
        var id = HEAP32[((ids)+(i*4))>>2];
        var query = GL.queries[id];
        if (!query) continue; // GL spec: "unused names in ids are ignored, as is the
name zero."
        GLctx['deleteQuery'](query);
        GL.queries[id] = null;
    }
}

function _glDeleteRenderbuffers(n, renderbuffers) {
    for (var i = 0; i < n; i++) {
        var id = HEAP32[((renderbuffers)+(i*4))>>2];
        var renderbuffer = GL.renderbuffers[id];
        if (!renderbuffer) continue; // GL spec: "glDeleteRenderbuffers silently
ignores 0s and names that do not correspond to existing renderbuffer objects".
        GLctx.deleteRenderbuffer(renderbuffer);
        renderbuffer.name = 0;
        GL.renderbuffers[id] = null;
    }
}

function _glDeleteSamplers(n, samplers) {
    for (var i = 0; i < n; i++) {
        var id = HEAP32[((samplers)+(i*4))>>2];
        var sampler = GL.samplers[id];
        if (!sampler) continue;
        GLctx['deleteSampler'](sampler);
        sampler.name = 0;
        GL.samplers[id] = null;
    }
}

function _glDeleteShader(id) {
    if (!id) return;
    var shader = GL.shaders[id];
    if (!shader) { // glDeleteShader actually signals an error when deleting a
nonexisting object, unlike some other GL delete functions.
        GL.recordError(0x501 /* GL_INVALID_VALUE */);
        return;
    }
    GLctx.deleteShader(shader);
    GL.shaders[id] = null;
}

function _glDeleteSync(id) {
    if (!id) return;
    var sync = GL.syncs[id];
    if (!sync) { // glDeleteSync signals an error when deleting a nonexisting object,
unlike some other GL delete functions.
        GL.recordError(0x501 /* GL_INVALID_VALUE */);

```

```

        return;
    }
    GLctx.deleteSync(sync);
    sync.name = 0;
    GL.syncs[id] = null;
}

function _glDeleteTextures(n, textures) {
    for (var i = 0; i < n; i++) {
        var id = HEAP32[((textures)+(i*4))>>2];
        var texture = GL.textures[id];
        if (!texture) continue; // GL spec: "glDeleteTextures silently ignores 0s and
names that do not correspond to existing textures".
        GLctx.deleteTexture(texture);
        texture.name = 0;
        GL.textures[id] = null;
    }
}

function _glDeleteVertexArrays(n, vaos) {
    for (var i = 0; i < n; i++) {
        var id = HEAP32[((vaos)+(i*4))>>2];
        GLctx['deleteVertexArray'](GL.vaos[id]);
        GL.vaos[id] = null;
    }
}

function _glDepthFunc(x0) { GLctx['depthFunc'](x0) }

function _glDepthMask(flag) {
    GLctx.depthMask(!flag);
}

function _glDetachShader(program, shader) {
    GLctx.detachShader(GL.programs[program], GL.shaders[shader]);
}

function _glDisable(x0) { GLctx['disable'](x0) }

function _glDisableVertexAttribArray(index) {
    var cb = GL.currentContext.clientBuffers[index];
    cb.enabled = false;
    GLctx.disableVertexAttribArray(index);
}

function _glDrawArrays(mode, first, count) {
    // bind any client-side buffers
    GL.preDrawHandleClientVertexAttribBindings(first + count);

    GLctx.drawArrays(mode, first, count);

    GL.postDrawHandleClientVertexAttribBindings();
}

function _glDrawArraysInstanced(mode, first, count, primcount) {

```

```

        GLctx['drawArraysInstanced'](mode, first, count, primcount);
    }

var tempFixedLengthArray = [];
function _glDrawBuffers(n, bufs) {

    var bufArray = tempFixedLengthArray[n];
    for (var i = 0; i < n; i++) {
        bufArray[i] = HEAP32[((bufs)+(i*4))>>2];
    }

    GLctx['drawBuffers'](bufArray);
}

function _glDrawElements(mode, count, type, indices) {
    var buf;
    if (!GLctx.currentElementArrayBufferBinding) {
        var size = GL.calcBufLength(1, type, 0, count);
        buf = GL.getTempIndexBuffer(size);
        GLctx.bindBuffer(0x8893 /*GL_ELEMENT_ARRAY_BUFFER*/, buf);
        GLctx.bufferSubData(0x8893 /*GL_ELEMENT_ARRAY_BUFFER*/,
                            0,
                            HEAPU8.subarray(indices, indices + size));
        // the index is now 0
        indices = 0;
    }

    // bind any client-side buffers
    GL.preDrawHandleClientVertexAttribBindings(count);

    GLctx.drawElements(mode, count, type, indices);

    GL.postDrawHandleClientVertexAttribBindings(count);

    if (!GLctx.currentElementArrayBufferBinding) {
        GLctx.bindBuffer(0x8893 /*GL_ELEMENT_ARRAY_BUFFER*/, null);
    }
}

function _glDrawElementsInstanced(mode, count, type, indices, primcount) {
    GLctx['drawElementsInstanced'](mode, count, type, indices, primcount);
}

function _glEnable(x0) { GLctx['enable'](x0) }

function _glEnableVertexAttribArray(index) {
    var cb = GL.currentThread.clientBuffers[index];
    cb.enabled = true;
    GLctx.enableVertexAttribArray(index);
}

function _glEndQuery(x0) { GLctx['endQuery'](x0) }

function _glFenceSync(condition, flags) {
    var sync = GLctx.fenceSync(condition, flags);
}

```

```

if (sync) {
    var id = GL.getId(GL.syncs);
    sync.name = id;
    GL.syncs[id] = sync;
    return id;
} else {
    return 0; // Failed to create a sync object
}

function _glFinish() { GLctx['finish']() }

function _glFlush() { GLctx['flush']() }

function emscriptenWebGLGetBufferBinding(target) {
    switch (target) {
        case 0x8892 /*GL_ARRAY_BUFFER*/: target = 0x8894 /*GL_ARRAY_BUFFER_BINDING*/;
break;
        case 0x8893 /*GL_ELEMENT_ARRAY_BUFFER*/: target = 0x8895 /*GL_ELEMENT_ARRAY_BUFFER_BINDING*/; break;
        case 0x88EB /*GL_PIXEL_PACK_BUFFER*/: target = 0x88ED /*GL_PIXEL_PACK_BUFFER_BINDING*/; break;
        case 0x88EC /*GL_PIXEL_UNPACK_BUFFER*/: target = 0x88EF /*GL_PIXEL_UNPACK_BUFFER_BINDING*/; break;
        case 0x8C8E /*GL_TRANSFORM_FEEDBACK_BUFFER*/: target = 0x8C8F /*GL_TRANSFORM_FEEDBACK_BUFFER_BINDING*/; break;
        case 0x8F36 /*GL_COPY_READ_BUFFER*/: target = 0x8F36 /*GL_COPY_READ_BUFFER_BINDING*/; break;
        case 0x8F37 /*GL_COPY_WRITE_BUFFER*/: target = 0x8F37 /*GL_COPY_WRITE_BUFFER_BINDING*/; break;
        case 0x8A11 /*GL_UNIFORM_BUFFER*/: target = 0x8A28 /*GL_UNIFORM_BUFFER_BINDING*/; break;
        // In default case, fall through and assume passed one of the _BINDING enums directly.
    }
    var buffer = GLctx.getParameter(target);
    if (buffer) return buffer.name|0;
    else return 0;
}

function emscriptenWebGLValidateMapBufferTarget(target) {
    switch (target) {
        case 0x8892: // GL_ARRAY_BUFFER
        case 0x8893: // GL_ELEMENT_ARRAY_BUFFER
        case 0x8F36: // GL_COPY_READ_BUFFER
        case 0x8F37: // GL_COPY_WRITE_BUFFER
        case 0x88EB: // GL_PIXEL_PACK_BUFFER
        case 0x88EC: // GL_PIXEL_UNPACK_BUFFER
        case 0x8C2A: // GL_TEXTURE_BUFFER
        case 0x8C8E: // GL_TRANSFORM_FEEDBACK_BUFFER
        case 0x8A11: // GL_UNIFORM_BUFFER
            return true;
        default:
            return false;
    }
}

```

```

        }

function _glFlushMappedBufferRange(target, offset, length) {
    if (!emscriptenWebGLValidateMapBufferTarget(target)) {
        GL.recordError(0x500/*GL_INVALID_ENUM*/);
        err('GL_INVALID_ENUM in glFlushMappedBufferRange');
        return;
    }

    var mapping = GL.mappedBuffers[emscriptenWebGLGetBufferBinding(target)];
    if (!mapping) {
        GL.recordError(0x502 /* GL_INVALID_OPERATION */);
        err('buffer was never mapped in glFlushMappedBufferRange');
        return;
    }

    if (!(mapping.access & 0x10)) {
        GL.recordError(0x502 /* GL_INVALID_OPERATION */);
        err('buffer was not mapped with GL_MAP_FLUSH_EXPLICIT_BIT in
glFlushMappedBufferRange');
        return;
    }
    if (offset < 0 || length < 0 || offset + length > mapping.length) {
        GL.recordError(0x501 /* GL_INVALID_VALUE */);
        err('invalid range in glFlushMappedBufferRange');
        return;
    }

    GLctx.bufferSubData(
        target,
        mapping.offset,
        HEAPU8.subarray(mapping.mem + offset, mapping.mem + offset + length));
}

function _glFramebufferRenderbuffer(target, attachment, renderbuffertarget,
renderbuffer) {
    GLctx.framebufferRenderbuffer(target, attachment, renderbuffertarget,
                                    GL.renderbuffers[renderbuffer]);
}

function _glFramebufferTexture2D(target, attachment, textarget, texture, level) {
    GLctx.framebufferTexture2D(target, attachment, textarget,
                                GL.textures[texture], level);
}

function _glFramebufferTextureLayer(target, attachment, texture, level, layer) {
    GLctx.framebufferTextureLayer(target, attachment, GL.textures[texture], level,
layer);
}

function _glFrontFace(x0) { GLctx['frontFace'](x0) }

function __glGenObject(n, buffers, createFunction, objectTable
) {
    for (var i = 0; i < n; i++) {
        var buffer = GLctx[createFunction]();
}

```

```

        var id = buffer && GL.getNewId(objectTable);
        if (buffer) {
            buffer.name = id;
            objectTable[id] = buffer;
        } else {
            GL.recordError(0x502 /* GL_INVALID_OPERATION */);
        }
        HEAP32[((buffers)+(i*4))>>2] = id;
    }
}

function _glGenBuffers(n, buffers) {
    __glGenObject(n, buffers, 'createBuffer', GL.buffers
);
}

function _glGenFramebuffers(n, ids) {
    __glGenObject(n, ids, 'createFramebuffer', GL.framebuffers
);
}

function _glGenQueries(n, ids) {
    __glGenObject(n, ids, 'createQuery', GL.queries
);
}

function _glGenRenderbuffers(n, renderbuffers) {
    __glGenObject(n, renderbuffers, 'createRenderbuffer', GL.renderbuffers
);
}

function _glGenSamplers(n, samplers) {
    __glGenObject(n, samplers, 'createSampler', GL.samplers
);
}

function _glGenTextures(n, textures) {
    __glGenObject(n, textures, 'createTexture', GL.textures
);
}

function _glGenVertexArrays(n, arrays) {
    __glGenObject(n, arrays, 'createVertexArray', GL.vaos
);
}

function _glGenerateMipmap(x0) { GLctx['generateMipmap'](x0) }

function _glGetActiveAttribOrUniform(funcName, program, index, bufSize, length,
size, type, name) {
    program = GL.programs[program];
    var info = GLctx[funcName](program, index);
    if (info) { // If an error occurs, nothing will be written to length, size and
type and name.
        var numBytesWrittenExclNull = name && stringToUTF8(info.name, name, bufSize);
        if (length) HEAP32[((length)>>2)] = numBytesWrittenExclNull;
    }
}

```

```

        if (size) HEAP32[((size)>>2)] = info.size;
        if (type) HEAP32[((type)>>2)] = info.type;
    }
}

function _glGetActiveAttrib(program, index, bufSize, length, size, type, name) {
    __glGetActiveAttribOrUniform('getActiveAttrib', program, index, bufSize, length,
size, type, name);
}

function _glGetActiveUniform(program, index, bufSize, length, size, type, name) {
    __glGetActiveAttribOrUniform('getActiveUniform', program, index, bufSize, length,
size, type, name);
}

function _glGetActiveUniformBlockName(program, uniformBlockIndex, bufSize, length,
uniformBlockName) {
    program = GL.programs[program];

    var result = GLctx['getActiveUniformBlockName'](program, uniformBlockIndex);
    if (!result) return; // If an error occurs, nothing will be written to
uniformBlockName or length.
    if (uniformBlockName && bufSize > 0) {
        var numBytesWrittenExclNull = stringToUTF8(result, uniformBlockName, bufSize);
        if (length) HEAP32[((length)>>2)] = numBytesWrittenExclNull;
    } else {
        if (length) HEAP32[((length)>>2)] = 0;
    }
}

function _glGetActiveUniformBlockiv(program, uniformBlockIndex, pname, params) {
    if (!params) {
        // GLES2 specification does not specify how to behave if params is a null
pointer. Since calling this function does not make sense
        // if params == null, issue a GL error to notify user about it.
        GL.recordError(0x501 /* GL_INVALID_VALUE */);
        return;
    }
    program = GL.programs[program];

    if (pname == 0x8A41 /* GL_UNIFORM_BLOCK_NAME_LENGTH */) {
        var name = GLctx['getActiveUniformBlockName'](program, uniformBlockIndex);
        HEAP32[((params)>>2)] = name.length+1;
        return;
    }

    var result = GLctx['getActiveUniformBlockParameter'](program, uniformBlockIndex,
pname);
    if (result === null) return; // If an error occurs, nothing should be written to
params.
    if (pname == 0x8A43 /*GL_UNIFORM_BLOCK_ACTIVE_UNIFORM_INDICES*/) {
        for (var i = 0; i < result.length; i++) {
            HEAP32[((params)+(i*4))>>2] = result[i];
        }
    } else {
        HEAP32[((params)>>2)] = result;
    }
}

```

```

        }
    }

    function _glGetActiveUniformsiv(program, uniformCount, uniformIndices, pname, params)
{
    if (!params) {
        // GLES2 specification does not specify how to behave if params is a null
        // pointer. Since calling this function does not make sense
        // if params == null, issue a GL error to notify user about it.
        GL.recordError(0x501 /* GL_INVALID_VALUE */);
        return;
    }
    if (uniformCount > 0 && uniformIndices == 0) {
        GL.recordError(0x501 /* GL_INVALID_VALUE */);
        return;
    }
    program = GL.programs[program];
    var ids = [];
    for (var i = 0; i < uniformCount; i++) {
        ids.push(HEAP32[((uniformIndices)+(i*4))>>2]);
    }

    var result = GLctx['getActiveUniforms'](program, ids, pname);
    if (!result) return; // GL spec: If an error is generated, nothing is written out
    to params.

    var len = result.length;
    for (var i = 0; i < len; i++) {
        HEAP32[((params)+(i*4))>>2] = result[i];
    }
}

function _glGetAttribLocation(program, name) {
    return GLctx.getAttribLocation(GL.programs[program], UTF8ToString(name));
}

function _glGetBufferSubData(target, offset, size, data) {
    if (!data) {
        // GLES2 specification does not specify how to behave if data is a null
        // pointer. Since calling this function does not make sense
        // if data == null, issue a GL error to notify user about it.
        GL.recordError(0x501 /* GL_INVALID_VALUE */);
        return;
    }
    GLctx['getBufferSubData'](target, offset, HEAPU8, data, size);
}

function _glGetError() {
    var error = GLctx.getError() || GL.lastError;
    GL.lastError = 0/*GL_NO_ERROR*/;
    return error;
}

function _glGetFramebufferAttachmentParameteriv(target, attachment, pname, params) {
    var result = GLctx.getFramebufferAttachmentParameter(target, attachment, pname);
}

```

```

    if (result instanceof WebGLRenderbuffer ||
        result instanceof WebGLTexture) {
        result = result.name | 0;
    }
    HEAP32[((params)>>2)] = result;
}

function readI53FromI64(ptr) {
    return HEAPU32[ptr>>2] + HEAP32[ptr+4>>2] * 4294967296;
}

function readI53FromU64(ptr) {
    return HEAPU32[ptr>>2] + HEAPU32[ptr+4>>2] * 4294967296;
}
function writeI53ToI64(ptr, num) {
    HEAPU32[ptr>>2] = num;
    HEAPU32[ptr+4>>2] = (num - HEAPU32[ptr>>2])/4294967296;
    var deserialized = (num >= 0) ? readI53FromU64(ptr) : readI53FromI64(ptr);
    if (deserialized != num) warnOnce('writeI53ToI64() out of range: serialized JS
Number ' + num + ' to Wasm heap as bytes lo=0x' + HEAPU32[ptr>>2].toString(16) + ',
hi=0x' + HEAPU32[ptr+4>>2].toString(16) + ', which deserializes back to ' +
deserialized + ' instead!');
}
function emscriptenWebGLGetIndexed(target, index, data, type) {
    if (!data) {
        // GLES2 specification does not specify how to behave if data is a null
pointer. Since calling this function does not make sense
        // if data == null, issue a GL error to notify user about it.
        GL.recordError(0x501 /* GL_INVALID_VALUE */);
        return;
    }
    var result = GLctx['getIndexedParameter'](target, index);
    var ret;
    switch (typeof result) {
        case 'boolean':
            ret = result ? 1 : 0;
            break;
        case 'number':
            ret = result;
            break;
        case 'object':
            if (result === null) {
                switch (target) {
                    case 0x8C8F: // TRANSFORM_FEEDBACK_BUFFER_BINDING
                    case 0x8A28: // UNIFORM_BUFFER_BINDING
                        ret = 0;
                        break;
                    default: {
                        GL.recordError(0x500); // GL_INVALID_ENUM
                        return;
                    }
                }
            } else if (result instanceof WebGLBuffer) {
                ret = result.name | 0;
            } else {

```

```

        GL.recordError(0x500); // GL_INVALID_ENUM
        return;
    }
    break;
default:
    GL.recordError(0x500); // GL_INVALID_ENUM
    return;
}

switch (type) {
case 1: writeI53ToI64(data, ret); break;
case 0: HEAP32[((data)>>2)] = ret; break;
case 2: HEAPF32[((data)>>2)] = ret; break;
case 4: HEAP8[((data)>>0)] = ret ? 1 : 0; break;
default: throw 'internal emscriptenWebGLGetIndexed() error, bad type: ' + type;
}
}

function _glGetIntegeri_v(target, index, data) {
    emscriptenWebGLGetIndexed(target, index, data, 0);
}

function emscriptenWebGLGet(name_, p, type) {
    // Guard against user passing a null pointer.
    // Note that GLES2 spec does not say anything about how passing a null pointer
    // should be treated.
    // Testing on desktop core GL 3, the application crashes on glGetIntegerv to a
    // null pointer, but
    // better to report an error instead of doing anything random.
    if (!p) {
        GL.recordError(0x501 /* GL_INVALID_VALUE */);
        return;
    }
    var ret = undefined;
    switch (name_) { // Handle a few trivial GLES values
        case 0x8DFA: // GL_SHADER_COMPILER
            ret = 1;
            break;
        case 0x8DF8: // GL_SHADER_BINARY_FORMATS
            if (type != 0 && type != 1) {
                GL.recordError(0x500); // GL_INVALID_ENUM
            }
            return; // Do not write anything to the out pointer, since no binary formats
        // are supported.
        case 0x87FE: // GL_NUM_PROGRAM_BINARY_FORMATS
        case 0x8DF9: // GL_NUM_SHADER_BINARY_FORMATS
            ret = 0;
            break;
        case 0x86A2: // GL_NUM_COMPRESSED_TEXTURE_FORMATS
            // WebGL doesn't have GL_NUM_COMPRESSED_TEXTURE_FORMATS (it's obsolete since
            // GL_COMPRESSED_TEXTURE_FORMATS returns a JS array that can be queried for length),
            // so implement it ourselves to allow C++ GLES2 code get the length.
            var formats = GLctx.getParameter(0x86A3 /*GL_COMPRESSED_TEXTURE_FORMATS*/);
            ret = formats ? formats.length : 0;
            break;
        case 0x826E: // GL_MAX_UNIFORM_LOCATIONS
    }
}
```

```

// This is an arbitrary limit, must be large enough to allow practical
// use, but small enough to still keep a range for automatic uniform
// locations, which get assigned numbers larger than this.
ret = 1048576;
break;

case 0x821D: // GL_NUM_EXTENSIONS
    if (GL.currentContext.version < 2) {
        GL.recordError(0x502 /* GL_INVALID_OPERATION */); // Calling GLES3/WebGL2
function with a GLES2/WebGL1 context
        return;
    }
    // .getSupportedExtensions() can return null if context is lost, so coerce to
empty array.
    var exts = GLctx.getSupportedExtensions() || [];
    ret = 2 * exts.length; // each extension is duplicated, first in unprefixed
WebGL form, and then a second time with "GL_" prefix.
    break;
case 0x821B: // GL_MAJOR_VERSION
case 0x821C: // GL_MINOR_VERSION
    if (GL.currentContext.version < 2) {
        GL.recordError(0x500); // GL_INVALID_ENUM
        return;
    }
    ret = name_ == 0x821B ? 3 : 0; // return version 3.0
    break;
}

if (ret === undefined) {
    var result = GLctx.getParameter(name_);
    switch (typeof result) {
        case "number":
            ret = result;
            break;
        case "boolean":
            ret = result ? 1 : 0;
            break;
        case "string":
            GL.recordError(0x500); // GL_INVALID_ENUM
            return;
        case "object":
            if (result === null) {
                // null is a valid result for some (e.g., which buffer is bound - perhaps
nothing is bound), but otherwise
                // can mean an invalid name_, which we need to report as an error
                switch (name_) {
                    case 0x8894: // ARRAY_BUFFER_BINDING
                    case 0x8B8D: // CURRENT_PROGRAM
                    case 0x8895: // ELEMENT_ARRAY_BUFFER_BINDING
                    case 0x8CA6: // FRAMEBUFFER_BINDING or DRAW_FRAMEBUFFER_BINDING
                    case 0x8CA7: // RENDERBUFFER_BINDING
                    case 0x8069: // TEXTURE_BINDING_2D
                    case 0x85B5: // WebGL 2 GL_VERTEX_ARRAY_BINDING, or WebGL 1 extension
OES_vertex_array_object GL_VERTEX_ARRAY_BINDING_OES
                    case 0x8F36: // COPY_READ_BUFFER_BINDING or COPY_READ_BUFFER
                }
            }
    }
}

```

```

        case 0x8F37: // COPY_WRITE_BUFFER_BINDING or COPY_WRITE_BUFFER
        case 0x88ED: // PIXEL_PACK_BUFFER_BINDING
        case 0x88EF: // PIXEL_UNPACK_BUFFER_BINDING
        case 0x8CAA: // READ_FRAMEBUFFER_BINDING
        case 0x8919: // SAMPLER_BINDING
        case 0x8C1D: // TEXTURE_BINDING_2D_ARRAY
        case 0x806A: // TEXTURE_BINDING_3D
        case 0x8E25: // TRANSFORM_FEEDBACK_BINDING
        case 0x8C8F: // TRANSFORM_FEEDBACK_BUFFER_BINDING
        case 0x8A28: // UNIFORM_BUFFER_BINDING
        case 0x8514: { // TEXTURE_BINDING_CUBE_MAP
            ret = 0;
            break;
        }
        default: {
            GL.recordError(0x500); // GL_INVALID_ENUM
            return;
        }
    }
} else if (result instanceof Float32Array ||
           result instanceof Uint32Array ||
           result instanceof Int32Array ||
           result instanceof Array) {
    for (var i = 0; i < result.length; ++i) {
        switch (type) {
            case 0: HEAP32[((p)+(i*4))>>2] = result[i]; break;
            case 2: HEAPF32[((p)+(i*4))>>2] = result[i]; break;
            case 4: HEAP8[((p)+(i))>>0] = result[i] ? 1 : 0; break;
        }
    }
    return;
} else {
    try {
        ret = result.name | 0;
    } catch(e) {
        GL.recordError(0x500); // GL_INVALID_ENUM
        err('GL_INVALID_ENUM in glGet' + type + 'v: Unknown object returned from WebGL getParameter(' + name_ + ')! (error: ' + e + ')');
        return;
    }
}
break;
default:
    GL.recordError(0x500); // GL_INVALID_ENUM
    err('GL_INVALID_ENUM in glGet' + type + 'v: Native code calling glGet' + type + 'v(' + name_ + ') and it returns ' + result + ' of type ' + typeof(result) + '!');
    return;
}
}

switch (type) {
    case 1: writeI53ToI64(p, ret); break;
    case 0: HEAP32[(p)>>2] = ret; break;
    case 2: HEAPF32[(p)>>2] = ret; break;
}

```

```

        case 4: HEAP8[((p)>>0)] = ret ? 1 : 0; break;
    }
}
function _glGetIntegerv(name_, p) {
    emscriptenWebGLGet(name_, p, 0);
}

function _glGetInternalformativ(target, internalformat, pname, bufSize, params) {
    if (bufSize < 0) {
        GL.recordError(0x501 /* GL_INVALID_VALUE */);
        return;
    }
    if (!params) {
        // GLES3 specification does not specify how to behave if values is a null
        // pointer. Since calling this function does not make sense
        // if values == null, issue a GL error to notify user about it.
        GL.recordError(0x501 /* GL_INVALID_VALUE */);
        return;
    }
    var ret = GLctx['getInternalformatParameter'](target, internalformat, pname);
    if (ret === null) return;
    for (var i = 0; i < ret.length && i < bufSize; ++i) {
        HEAP32[((params)+(i*4))>>2] = ret[i];
    }
}

function _glGetProgramBinary(program, bufSize, length, binaryFormat, binary) {
    GL.recordError(0x502/*GL_INVALID_OPERATION*/);
}

function _glGetProgramInfoLog(program, maxLength, length, infoLog) {
    var log = GLctx.getProgramInfoLog(GL.programs[program]);
    if (log === null) log = '(unknown error)';
    var numBytesWrittenExclNull = (maxLength > 0 && infoLog) ? stringToUTF8(log,
infoLog, maxLength) : 0;
    if (length) HEAP32[((length)>>2)] = numBytesWrittenExclNull;
}

function _glGetProgramiv(program, pname, p) {
    if (!p) {
        // GLES2 specification does not specify how to behave if p is a null pointer.
        // Since calling this function does not make sense
        // if p == null, issue a GL error to notify user about it.
        GL.recordError(0x501 /* GL_INVALID_VALUE */);
        return;
    }

    if (program >= GL.counter) {
        GL.recordError(0x501 /* GL_INVALID_VALUE */);
        return;
    }

    program = GL.programs[program];

    if (pname == 0x8B84) { // GL_INFO_LOG_LENGTH

```

```

var log = GLctx.getProgramInfoLog(program);
if (log === null) log = '(unknown error)';
HEAP32[((p)>>2)] = log.length + 1;
} else if (pname == 0x8B87 /* GL_ACTIVE_UNIFORM_MAX_LENGTH */) {
    if (!program.maxUniformLength) {
        for (var i = 0; i < GLctx.getProgramParameter(program,
0x8B86/*GL_ACTIVE_UNIFORMS*/); ++i) {
            program.maxUniformLength = Math.max(program.maxUniformLength,
GLctx.getActiveUniform(program, i).name.length+1);
        }
    }
    HEAP32[((p)>>2)] = program.maxUniformLength;
} else if (pname == 0x8B8A /* GL_ACTIVE_ATTRIBUTE_MAX_LENGTH */) {
    if (!program.maxAttributeLength) {
        for (var i = 0; i < GLctx.getProgramParameter(program,
0x8B89/*GL_ACTIVE_ATTRIBUTES*/); ++i) {
            program.maxAttributeLength = Math.max(program.maxAttributeLength,
GLctx.getActiveAttrib(program, i).name.length+1);
        }
    }
    HEAP32[((p)>>2)] = program.maxAttributeLength;
} else if (pname == 0x8A35 /* GL_ACTIVE_UNIFORM_BLOCK_MAX_NAME_LENGTH */) {
    if (!program.maxUniformBlockNameLength) {
        for (var i = 0; i < GLctx.getProgramParameter(program,
0x8A36/*GL_ACTIVE_UNIFORM_BLOCKS*/); ++i) {
            program.maxUniformBlockNameLength =
Math.max(program.maxUniformBlockNameLength, GLctx.getActiveUniformBlockName(program,
i).length+1);
        }
    }
    HEAP32[((p)>>2)] = program.maxUniformBlockNameLength;
} else {
    HEAP32[((p)>>2)] = GLctx.getProgramParameter(program, pname);
}
}

function _glGetQueryObjectuiv(id, pname, params) {
    if (!params) {
        // GLES2 specification does not specify how to behave if params is a null
pointer. Since calling this function does not make sense
        // if p == null, issue a GL error to notify user about it.
        GL.recordError(0x501 /* GL_INVALID_VALUE */);
        return;
    }
    var query = GL.queries[id];
    var param = GLctx['getQueryParameter'](query, pname);
    var ret;
    if (typeof param == 'boolean') {
        ret = param ? 1 : 0;
    } else {
        ret = param;
    }
    HEAP32[((params)>>2)] = ret;
}

function _glGetQueryiv(target, pname, params) {

```

```

if (!params) {
    // GLES2 specification does not specify how to behave if params is a null
    // pointer. Since calling this function does not make sense
    // if p == null, issue a GL error to notify user about it.
    GL.recordError(0x501 /* GL_INVALID_VALUE */);
    return;
}
HEAP32[((params)>>2)] = GLctx['getQuery'](target, pname);
}

function _glGetRenderbufferParameteriv(target, pname, params) {
if (!params) {
    // GLES2 specification does not specify how to behave if params is a null
    // pointer. Since calling this function does not make sense
    // if params == null, issue a GL error to notify user about it.
    GL.recordError(0x501 /* GL_INVALID_VALUE */);
    return;
}
HEAP32[((params)>>2)] = GLctx.getRenderbufferParameter(target, pname);
}

function _glGetShaderInfoLog(shader, maxLength, length, infoLog) {
var log = GLctx.getShaderInfoLog(GL.shaders[shader]);
if (log === null) log = '(unknown error)';
var numBytesWrittenExclNull = (maxLength > 0 && infoLog) ? stringToUTF8(log,
infoLog, maxLength) : 0;
if (length) HEAP32[((length)>>2)] = numBytesWrittenExclNull;
}

function _glGetShaderPrecisionFormat(shaderType, precisionType, range, precision) {
var result = GLctx.getShaderPrecisionFormat(shaderType, precisionType);
HEAP32[((range)>>2)] = result.rangeMin;
HEAP32[((range)+(4))>>2)] = result.rangeMax;
HEAP32[((precision)>>2)] = result.precision;
}

function _glGetShaderSource(shader, bufSize, length, source) {
var result = GLctx.getShaderSource(GL.shaders[shader]);
if (!result) return; // If an error occurs, nothing will be written to length or
source.
var numBytesWrittenExclNull = (bufSize > 0 && source) ? stringToUTF8(result,
source, bufSize) : 0;
if (length) HEAP32[((length)>>2)] = numBytesWrittenExclNull;
}

function _glGetShaderiv(shader, pname, p) {
if (!p) {
    // GLES2 specification does not specify how to behave if p is a null pointer.
    // Since calling this function does not make sense
    // if p == null, issue a GL error to notify user about it.
    GL.recordError(0x501 /* GL_INVALID_VALUE */);
    return;
}
if (pname == 0x8B84) { // GL_INFO_LOG_LENGTH
    var log = GLctx.getShaderInfoLog(GL.shaders[shader]);
}
}

```

```

    if (log === null) log = '(unknown error)';
    // The GLES2 specification says that if the shader has an empty info log,
    // a value of 0 is returned. Otherwise the log has a null char appended.
    // (An empty string is falsey, so we can just check that instead of
    // looking at log.length.)
    var logLength = log ? log.length + 1 : 0;
    HEAP32[((p)>>2)] = logLength;
} else if (pname == 0x8B88) { // GL_SHADER_SOURCE_LENGTH
    var source = GLctx.getShaderSource(GL.shaders[shader]);
    // source may be a null, or the empty string, both of which are falsey
    // values that we report a 0 length for.
    var sourceLength = source ? source.length + 1 : 0;
    HEAP32[((p)>>2)] = sourceLength;
} else {
    HEAP32[((p)>>2)] = GLctx.getShaderParameter(GL.shaders[shader], pname);
}
}

function _glGetString(name_) {
    var ret = GL.stringCache[name_];
    if (!ret) {
        switch (name_) {
            case 0x1F03 /* GL_EXTENSIONS */:
                var exts = GLctx.getSupportedExtensions() || [];
                // .getSupportedExtensions() can return null if context is lost, so coerce to empty array.
                exts = exts.concat(exts.map(function(e) { return "GL_" + e; }));
                ret = stringToNewUTF8(exts.join(' '));
                break;
            case 0x1F00 /* GL_VENDOR */:
            case 0x1F01 /* GL_RENDERER */:
            case 0x9245 /* UNMASKED_VENDOR_WEBGL */:
            case 0x9246 /* UNMASKED_RENDERER_WEBGL */:
                var s = GLctx.getParameter(name_);
                if (!s) {
                    GL.recordError(0x500/*GL_INVALID_ENUM*/);
                }
                ret = s && stringToNewUTF8(s);
                break;

            case 0x1F02 /* GL_VERSION */:
                var glVersion = GLctx.getParameter(0x1F02 /*GL_VERSION*/);
                // return GLES version string corresponding to the version of the WebGL
                context
                if (GL.currentContext.version >= 2) glVersion = 'OpenGL ES 3.0 (' +
glVersion + ')';
                else
                {
                    glVersion = 'OpenGL ES 2.0 (' + glVersion + ')';
                }
                ret = stringToNewUTF8(glVersion);
                break;
            case 0x8B8C /* GL_SHADING_LANGUAGE_VERSION */:
                var glslVersion = GLctx.getParameter(0x8B8C
/*GL_SHADING_LANGUAGE_VERSION*/);
                // extract the version number 'N.M' from the string 'WebGL GLSL ES N.M ...'
        }
    }
}

```

```

        var ver_re = /^WebGL GLSL ES ([0-9]\.[0-9][0-9]?) (?:$| .*)/;
        var ver_num = glslVersion.match(ver_re);
        if (ver_num !== null) {
            if (ver_num[1].length == 3) ver_num[1] = ver_num[1] + '0'; // ensure
minor version has 2 digits
            glslVersion = 'OpenGL ES GLSL ES ' + ver_num[1] + ' (' + glslVersion +
')';
        }
        ret = stringToNewUTF8(glslVersion);
        break;
    default:
        GL.recordError(0x500/*GL_INVALID_ENUM*/);
        // fall through
    }
    GL.stringCache[name_] = ret;
}
return ret;
}

function _glGetStringi(name, index) {
    if (GL.currentContext.version < 2) {
        GL.recordError(0x502 /* GL_INVALID_OPERATION */); // Calling GLES3/WebGL2
function with a GLES2/WebGL1 context
        return 0;
    }
    var stringiCache = GL.stringiCache[name];
    if (stringiCache) {
        if (index < 0 || index >= stringiCache.length) {
            GL.recordError(0x501/*GL_INVALID_VALUE*/);
            return 0;
        }
        return stringiCache[index];
    }
    switch (name) {
        case 0x1F03 /* GL_EXTENSIONS */:
            var exts = GLctx.getSupportedExtensions() || []; // .getSupportedExtensions()
can return null if context is lost, so coerce to empty array.
            exts = exts.concat(exts.map(function(e) { return "GL_" + e; }));
            exts = exts.map(function(e) { return stringToNewUTF8(e); });

            stringiCache = GL.stringiCache[name] = exts;
            if (index < 0 || index >= stringiCache.length) {
                GL.recordError(0x501/*GL_INVALID_VALUE*/);
                return 0;
            }
            return stringiCache[index];
        default:
            GL.recordError(0x500/*GL_INVALID_ENUM*/);
            return 0;
    }
}

function _glGetTexParameteriv(target, pname, params) {
    if (!params) {

```

```

    // GLES2 specification does not specify how to behave if params is a null
    // pointer. Since calling this function does not make sense
    // if p == null, issue a GL error to notify user about it.
    GL.recordError(0x501 /* GL_INVALID_VALUE */);
    return;
}
HEAP32[((params)>>2)] = GLctx.getTexParameter(target, pname);
}

function _glGetUniformBlockIndex(program, uniformBlockName) {
    return GLctx['getUniformBlockIndex'](GL.programs[program],
UTF8ToString(uniformBlockName));
}

function _glGetUniformIndices(program, uniformCount, uniformNames, uniformIndices) {
    if (!uniformIndices) {
        // GLES2 specification does not specify how to behave if uniformIndices is a
        // null pointer. Since calling this function does not make sense
        // if uniformIndices == null, issue a GL error to notify user about it.
        GL.recordError(0x501 /* GL_INVALID_VALUE */);
        return;
    }
    if (uniformCount > 0 && (uniformNames == 0 || uniformIndices == 0)) {
        GL.recordError(0x501 /* GL_INVALID_VALUE */);
        return;
    }
    program = GL.programs[program];
    var names = [];
    for (var i = 0; i < uniformCount; i++)
        names.push(UTF8ToString(HEAP32[((uniformNames)+(i*4))>>2]));

    var result = GLctx['getUniformIndices'](program, names);
    if (!result) return; // GL spec: If an error is generated, nothing is written out
    to uniformIndices.

    var len = result.length;
    for (var i = 0; i < len; i++) {
        HEAP32[((uniformIndices)+(i*4))>>2] = result[i];
    }
}

/** @noinline */
function webglGetLeftBracePos(name) {
    return name.slice(-1) == ']' && name.lastIndexOf('[');
}
function webglPrepareUniformLocationsBeforeFirstUse(program) {
    var uniformLocsById = program.uniformLocsById, // Maps GLuint ->
WebGLUniformLocation
    uniformSizeAndIdsByName = program.uniformSizeAndIdsByName, // Maps name ->
[uniform array length, GLuint]
    i, j;

    // On the first time invocation of glGetUniformLocation on this shader program:
    // initialize cache data structures and discover which uniforms are arrays.
    if (!uniformLocsById) {
        // maps GLint integer locations to WebGLUniformLocation

```

```

program.uniformLocsById = uniformLocsById = {};
// maps integer locations back to uniform name strings, so that we can lazily
fetch uniform array locations
program.uniformArrayNamesById = {};

for (i = 0; i < GLctx.getProgramParameter(program,
0x8B86/*GL_ACTIVE_UNIFORMS*/); ++i) {
    var u = GLctx.getActiveUniform(program, i);
    var nm = u.name;
    var sz = u.size;
    var lb = webglGetLeftBracePos(nm);
    var arrayName = lb > 0 ? nm.slice(0, lb) : nm;

    // Acquire the preset location from the explicit uniform location if one was
specified, or
    // programmatically assign a new one if not.
    var id = uniformSizeAndIdsByName[arrayName] ?
uniformSizeAndIdsByName[arrayName][1] : program.uniformIdCounter;
    program.uniformIdCounter = Math.max(id + sz, program.uniformIdCounter);
    // Eagerly get the location of the uniformArray[0] base element.
    // The remaining indices >0 will be left for lazy evaluation to
    // improve performance. Those may never be needed to fetch, if the
    // application fills arrays always in full starting from the first
    // element of the array.
    uniformSizeAndIdsByName[arrayName] = [sz, id];

    // Store placeholder integers in place that highlight that these
    // >0 index locations are array indices pending population.
    for(j = 0; j < sz; ++j) {
        uniformLocsById[id] = j;
        program.uniformArrayNamesById[id++] = arrayName;
    }
}
}

function _glGetUniformLocation(program, name) {

name = UTF8ToString(name);

if (program = GL.programs[program]) {
    webglPrepareUniformLocationsBeforeFirstUse(program);
    var uniformLocsById = program.uniformLocsById; // Maps GLuint ->
WebGLUniformLocation
    var arrayIndex = 0;
    var uniformBaseName = name;

    // Invariant: when populating integer IDs for uniform locations, we must
maintain the precondition that
    // arrays reside in contiguous addresses, i.e. for a 'vec4 colors[10];',
colors[4] must be at location colors[0]+4.
    // However, user might call glGetUniformLocation(program, "colors") for an
array, so we cannot discover based on the user
    // input arguments whether the uniform we are dealing with is an array. The
only way to discover which uniforms are arrays
    // is to enumerate over all the active uniforms in the program.
}
}

```

```

var leftBrace = webglGetLeftBracePos(name);

// If user passed an array accessor "[index]", parse the array index off the
accessor.
if (leftBrace > 0) {
    arrayIndex = parseInt(name.slice(leftBrace + 1)) >>> 0; // "index]", coerce
    parseInt('[') with >>>0 to treat "foo[]" as "foo[0]" and foo[-1] as unsigned out-of-
    bounds.
    uniformBaseName = name.slice(0, leftBrace);
}

// Have we cached the location of this uniform before?
var sizeAndId = program.uniformSizeAndIdsByName[uniformBaseName]; // A pair
[array length, GLint of the uniform location]

// If an uniform with this name exists, and if its index is within the array
limits (if it's even an array),
// query the WebGLlocation, or return an existing cached location.
if (sizeAndId && arrayIndex < sizeAndId[0]) {
    arrayIndex += sizeAndId[1]; // Add the base location of the uniform to the
array index offset.
    if ((uniformLocsById[arrayIndex] = uniformLocsById[arrayIndex] ||
GLctx.getUniformLocation(program, name))) {
        return arrayIndex;
    }
}
else {
    // N.b. we are currently unable to distinguish between GL program IDs that
never existed vs GL program IDs that have been deleted,
    // so report GL_INVALID_VALUE in both cases.
    GL.recordError(0x501 /*GL_INVALID_VALUE*/);
}
return -1;
}

function webglGetUniformLocation(location) {
var p = GLctx.currentProgram;

if (p) {
    var webglLoc = p.uniformLocsById[location];
    // p.uniformLocsById[location] stores either an integer, or a
WebGLUniformLocation.

    // If an integer, we have not yet bound the location, so do it now. The integer
value specifies the array index
    // we should bind to.
    if (typeof webglLoc == 'number') {
        p.uniformLocsById[location] = webglLoc = GLctx.getUniformLocation(p,
p.uniformArrayNamesById[location] + (webglLoc > 0 ? '[' + webglLoc + ']' : ''));
    }
    // Else an already cached WebGLUniformLocation, return it.
    return webglLoc;
} else {
    GL.recordError(0x502/*GL_INVALID_OPERATION*/);
}
}

```

```

        }
    }
/** @suppress{checkTypes} */
function emscriptenWebGLGetUniform(program, location, params, type) {
    if (!params) {
        // GLES2 specification does not specify how to behave if params is a null
        // pointer. Since calling this function does not make sense
        // if params == null, issue a GL error to notify user about it.
        GL.recordError(0x501 /* GL_INVALID_VALUE */);
        return;
    }
    program = GL.programs[program];
    webglPrepareUniformLocationsBeforeFirstUse(program);
    var data = GLctx.getUniform(program, webglGetUniformLocation(location));
    if (typeof data == 'number' || typeof data == 'boolean') {
        switch (type) {
            case 0: HEAP32[((params)>>2)] = data; break;
            case 2: HEAPF32[((params)>>2)] = data; break;
        }
    } else {
        for (var i = 0; i < data.length; i++) {
            switch (type) {
                case 0: HEAP32[((params)+(i*4))>>2] = data[i]; break;
                case 2: HEAPF32[((params)+(i*4))>>2] = data[i]; break;
            }
        }
    }
}
function _glGetUniformiv(program, location, params) {
    emscriptenWebGLGetUniform(program, location, params, 0);
}

/** @suppress{checkTypes} */
function emscriptenWebGLGetVertexAttrib(index, pname, params, type) {
    if (!params) {
        // GLES2 specification does not specify how to behave if params is a null
        // pointer. Since calling this function does not make sense
        // if params == null, issue a GL error to notify user about it.
        GL.recordError(0x501 /* GL_INVALID_VALUE */);
        return;
    }
    if (GL.currentContext.clientBuffers[index].enabled) {
        err("glGetVertexAttrib*v on client-side array: not supported, bad data
returned");
    }
    var data = GLctx.getVertexAttrib(index, pname);
    if (pname == 0x889F/*VERTEX_ATTRIB_ARRAY_BUFFER_BINDING*/) {
        HEAP32[((params)>>2)] = data && data["name"];
    } else if (typeof data == 'number' || typeof data == 'boolean') {
        switch (type) {
            case 0: HEAP32[((params)>>2)] = data; break;
            case 2: HEAPF32[((params)>>2)] = data; break;
            case 5: HEAP32[((params)>>2)] = Math.fround(data); break;
        }
    } else {

```

```

        for (var i = 0; i < data.length; i++) {
            switch (type) {
                case 0: HEAP32[((params)+(i*4))>>2] = data[i]; break;
                case 2: HEAPF32[((params)+(i*4))>>2] = data[i]; break;
                case 5: HEAP32[((params)+(i*4))>>2] = Math.fround(data[i]); break;
            }
        }
    }
}

function _glGetVertexAttribiv(index, pname, params) {
    // N.B. This function may only be called if the vertex attribute was specified
    // using the function glVertexAttrib*f(),
    // otherwise the results are undefined. (GLES3 spec 6.1.12)
    emscriptenWebGLGetVertexAttrib(index, pname, params, 5);
}

function _glInvalidateFramebuffer(target, numAttachments, attachments) {
    var list = tempFixedLengthArray[numAttachments];
    for (var i = 0; i < numAttachments; i++) {
        list[i] = HEAP32[((attachments)+(i*4))>>2];
    }

    GLctx['invalidateFramebuffer'](target, list);
}

function _glIsEnabled(x0) { return GLctx['isEnabled'](x0) }

function _glIsVertexArray(array) {

    var vao = GL.vaos[array];
    if (!vao) return 0;
    return GLctx['isVertexArray'](vao);
}

function _glLinkProgram(program) {
    program = GL.programs[program];
    GLctx.linkProgram(program);
    // Invalidate earlier computed uniform->ID mappings, those have now become stale
    program.uniformLocsById = 0; // Mark as null-like so that glGetUniformLocation()
knows to populate this again.
    program.uniformSizeAndIdsByName = {};

    // Collect explicit uniform locations from the vertex and fragment shaders.
    [program['vs'], program['fs']].forEach(function(s) {
        Object.keys(s.explicitUniformLocations).forEach(function(shaderLocation) {
            var loc = s.explicitUniformLocations[shaderLocation];
            // Record each explicit uniform location temporarily as a non-array uniform
            // with size=1. This is not true, but on the first glGetUniformLocation()
call
            // the array sizes will get populated to correct sizes.
            program.uniformSizeAndIdsByName[shaderLocation] = [1, loc];

            // Make sure we will never automatically assign locations within the range
            // used for explicit layout(location=x) variables.
            program.uniformIdCounter = Math.max(program.uniformIdCounter, loc + 1);
        });
    });
}

```

```

        });
    });

    function copyKeys(dst, src) {
        Object.keys(src).forEach(function(key) {
            dst[key] = src[key];
        });
    }
    // Collect sampler and ubo binding locations from the vertex and fragment
    // shaders.
    program.explicitUniformBindings = {};
    program.explicitSamplerBindings = {};
    [program['vs'], program['fs']].forEach(function(s) {
        copyKeys(program.explicitUniformBindings, s.explicitUniformBindings);
        copyKeys(program.explicitSamplerBindings, s.explicitSamplerBindings);
    });
    // Record that we need to apply these explicit bindings when glUseProgram() is
    // first called on this program.
    program.explicitProgramBindingsApplied = 0;
}

function _glMapBufferRange(target, offset, length, access) {
    if (access != 0x1A && access != 0xA) {
        err("glMapBufferRange is only supported when access is
MAP_WRITE|INVALIDATE_BUFFER");
        return 0;
    }

    if (!emscriptenWebGLValidateMapBufferTarget(target)) {
        GL.recordError(0x500/*GL_INVALID_ENUM*/);
        err('GL_INVALID_ENUM in glMapBufferRange');
        return 0;
    }

    var mem = _malloc(length);
    if (!mem) return 0;

    GL.mappedBuffers[emscriptenWebGLGetBufferBinding(target)] = {
        offset: offset,
        length: length,
        mem: mem,
        access: access,
    };
    return mem;
}

function _glPixelStorei(pname, param) {
    if (pname == 0xCF5 /* GL_UNPACK_ALIGNMENT */) {
        GL.unpackAlignment = param;
    }
    GLctx.pixelStorei(pname, param);
}

function _glPolygonOffset(x0, x1) { GLctx['polygonOffset'](x0, x1) }

```

```

function _glProgramBinary(program, binaryFormat, binary, length) {
    GL.recordError(0x500/*GL_INVALID_ENUM*/);
}

function _glProgramParameteri(program, pname, value) {
    GL.recordError(0x500/*GL_INVALID_ENUM*/);
}

function _glReadBuffer(x0) { GLctx['readBuffer'](x0) }

function computeUnpackAlignedImageSize(width, height, sizePerPixel, alignment) {
    function roundedToNextMultipleOf(x, y) {
        return (x + y - 1) & -y;
    }
    var plainRowSize = width * sizePerPixel;
    var alignedRowSize = roundedToNextMultipleOf(plainRowSize, alignment);
    return height * alignedRowSize;
}

function __colorChannelsInGlTextureFormat(format) {
    // Micro-optimizations for size: map format to size by subtracting smallest enum
    // value (0x1902) from all values first.
    // Also omit the most common size value (1) from the list, which is assumed by
    formats not on the list.
    var colorChannels = {
        // 0x1902 /* GL_DEPTH_COMPONENT */ - 0x1902: 1,
        // 0x1906 /* GL_ALPHA */ - 0x1902: 1,
        5: 3,
        6: 4,
        // 0x1909 /* GL_LUMINANCE */ - 0x1902: 1,
        8: 2,
        29502: 3,
        29504: 4,
        // 0x1903 /* GL_RED */ - 0x1902: 1,
        26917: 2,
        26918: 2,
        // 0x8D94 /* GL_RED_INTEGER */ - 0x1902: 1,
        29846: 3,
        29847: 4
    };
    return colorChannels[format - 0x1902] || 1;
}

function heapObjectForWebGLType(type) {
    // Micro-optimization for size: Subtract lowest GL enum number (0x1400/* GL_BYTE */
    // from type to compare
    // smaller values for the heap, for shorter generated code size.
    // Also the type HEAPU16 is not tested for explicitly, but any unrecognized type
    will return out HEAPU16.
    // (since most types are HEAPU16)
    type -= 0x1400;
    if (type == 0) return HEAP8;

    if (type == 1) return HEAPU8;
}

```

```

    if (type == 2) return HEAP16;
    if (type == 4) return HEAP32;
    if (type == 6) return HEAPF32;

    if (type == 5
        || type == 28922
        || type == 28520
        || type == 30779
        || type == 30782
    )
        return HEAPU32;

    return HEAPU16;
}

function heapAccessShiftForWebGLHeap(heap) {
    return 31 - Math.clz32(heap.BYTES_PER_ELEMENT);
}
function emscriptenWebGLGetTexPixelData(type, format, width, height, pixels,
internalFormat) {
    var heap = heapObjectForWebGLType(type);
    var shift = heapAccessShiftForWebGLHeap(heap);
    var byteSize = 1<<shift;
    var sizePerPixel = __colorChannelsInGlTextureFormat(format) * byteSize;
    var bytes = computeUnpackAlignedImageSize(width, height, sizePerPixel,
GL.unpackAlignment);
    return heap.subarray(pixels >> shift, pixels + bytes >> shift);
}
function _glReadPixels(x, y, width, height, format, type, pixels) {
    if (GL.currentContext.version >= 2) { // WebGL 2 provides new garbage-free entry
points to call to WebGL. Use those always when possible.
        if (GLctx.currentPixelPackBufferBinding) {
            GLctx.readPixels(x, y, width, height, format, type, pixels);
        } else {
            var heap = heapObjectForWebGLType(type);
            GLctx.readPixels(x, y, width, height, format, type, heap, pixels >>
heapAccessShiftForWebGLHeap(heap));
        }
        return;
    }
    var pixelData = emscriptenWebGLGetTexPixelData(type, format, width, height,
pixels, format);
    if (!pixelData) {
        GL.recordError(0x500/*GL_INVALID_ENUM*/);
        return;
    }
    GLctx.readPixels(x, y, width, height, format, type, pixelData);
}

function _glRenderbufferStorage(x0, x1, x2, x3) { GLctx['renderbufferStorage'](x0,
x1, x2, x3) }

```

```

function _glRenderbufferStorageMultisample(x0, x1, x2, x3, x4) {
    GLctx['renderbufferStorageMultisample'](x0, x1, x2, x3, x4) }

function _glSamplerParameteri(sampler, pname, param) {
    GLctx['samplerParameteri'](GL.samplers[sampler], pname, param);
}

function _glScissor(x0, x1, x2, x3) { GLctx['scissor'](x0, x1, x2, x3) }

function find_closing_parens_index(arr, i, opening='(', closing=')') {
    for(var nesting = 0; i < arr.length; ++i) {
        if (arr[i] == opening) ++nesting;
        if (arr[i] == closing && --nesting == 0) {
            return i;
        }
    }
}

function preprocess_c_code(code, defs = {}) {
    var i = 0, // iterator over the input string
        len = code.length, // cache input length
        out = '', // generates the preprocessed output string
        stack = [1]; // preprocessing stack (state of active/inactive #ifdef/#else
blocks we are currently inside)
    // a mapping 'symbolname' -> function(args) which evaluates the given cpp macro,
e.g. #define FOO(x) x+10.
    defs['defined'] = (args) => { // built-in "#if defined(x)"" macro.
        assert(args.length == 1);
        return defs[args[0]] ? 1 : 0;
    };

    // Returns true if str[i] is whitespace.
    function isWhitespace(str, i) {
        return !(str.charCodeAt(i) > 32); // Compare as negation to treat end-of-string
undefined as whitespace
    }

    // Returns index to the next whitespace character starting at str[i].
    function nextWhitespace(str, i) {
        while(!isWhitespace(str, i)) ++i;
        return i;
    }

    // Returns an integer ID classification of the character at str[idx], used for
tokenization purposes.
    function classifyChar(str, idx) {
        var cc = str.charCodeAt(idx);
        assert(!(cc > 127), "Only 7-bit ASCII can be used in preprocessor
#endif/#define statements!");
        if (cc > 32) {
            if (cc < 48) return 1; // an operator symbol, any of !#$%&'()*+,.-./
            if (cc < 58) return 2; // a number 0123456789
            if (cc < 65) return 1; // an operator symbol, any of :;<=>?@
            if (cc < 91 || cc == 95/*_*/) return 3; // a character, any of A-Z or _
            if (cc < 97) return 1; // an operator symbol, any of [\]^`_
            if (cc < 123) return 3; // a character, any of a-z
            return 1; // an operator symbol, any of {|}~
        }
    }
}

```

```

        }
        return cc < 33 ? 0 : 4; // 0=whitespace, 4=end-of-string
    }

    // Returns a tokenized array of the given string expression, i.e. "FOO > BAR && BAZ" -> ["FOO", ">", "BAR", "&&", "BAZ"]
    // Optionally keeps whitespace as tokens to be able to reconstruct the original input string.
    function tokenize(exprString, keepWhitespace) {
        var out = [], len = exprString.length;
        for(var i = 0; i <= len; ++i) {
            var kind = classifyChar(exprString, i);
            if (kind == 2/*0-9*/ || kind == 3/*a-z*/) { // a character or a number
                for(var j = i+1; j <= len; ++j) {
                    var kind2 = classifyChar(exprString, j);
                    if (kind2 != kind && (kind2 != 2/*0-9*/ || kind != 3/*a-z*/)) { // parse
number sequence "423410", and identifier sequence "FOO32BAR"
                        out.push(exprString.substring(i, j));
                        i = j-1;
                        break;
                    }
                }
            } else if (kind == 1/*operator symbol*/) {
                // Lookahead for two-character operators.
                var op2 = exprString.substr(i, 2);
                if (['<=', '>=', '==', '!=', '&&', '||'].includes(op2)) {
                    out.push(op2);
                    ++i;
                } else {
                    out.push(exprString[i]);
                }
            }
        }
        return out;
    }

    // Expands preprocessing macros on substring str[lineStart...lineEnd]
    function expandMacros(str, lineStart, lineEnd) {
        if (lineEnd === undefined) lineEnd = str.length;
        var len = str.length;
        var out = '';
        for(var i = lineStart; i < lineEnd; ++i) {
            var kind = classifyChar(str, i);
            if (kind == 3/*a-z*/) {
                for(var j = i + 1; j <= lineEnd; ++j) {
                    var kind2 = classifyChar(str, j);
                    if (kind2 != 2/*0-9*/ && kind2 != 3/*a-z*/) {
                        var symbol = str.substring(i, j);
                        var pp = defs[symbol];
                        if (pp) {
                            var expanded = str.substring(lineStart, i);
                            if (pp.length && str[j] == '(') { // Expanding a macro? (#define
FOO(X) ...
                                var closeParens = find_closing_parens_index(str, j);
                                assert(str[closeParens] == ')');
                            }
                        }
                    }
                }
            }
        }
        return out;
    }
}

```

```

        expanded += pp(str.substring(j+1, closeParens).split(',') + str.substring(closeParens+1, lineEnd));
    } else { // Expanding a non-macro (#define FOO BAR)
        expanded += pp() + str.substring(j, lineEnd);
    }
    return expandMacros(expanded, 0);
} else {
    out += symbol;
    i = j-1;
    break;
}
}
}
}
else {
    out += str[i];
}
}
return out;
}

// Given a token list e.g. ['2', '>', '1'], returns a function that evaluates
that token list.
function buildExprTree(tokens) {
    // Consume tokens array into a function tree until the tokens array is
exhausted
    // to a single root node that evaluates it.
    while (tokens.length > 1 || typeof tokens[0] != 'function') {
        tokens = (function(tokens) {
            // Find the index 'i' of the operator we should evaluate next:
            var i, j, p, operatorAndPriority = -2;
            for(j = 0; j < tokens.length; ++j) {
                if ((p = ['*', '/', '+', '-', '!', '<', '<=', '>', '>=', '==', '!=',
'&&', '||', '('].indexOf(tokens[j])) > operatorAndPriority) {
                    i = j;
                    operatorAndPriority = p;
                }
            }

            if (operatorAndPriority == 13 /* parens '(' */) {
                // Find the closing parens position
                var j = find_closing_parens_index(tokens, i);
                if (j) {
                    tokens.splice(i, j+1-i, buildExprTree(tokens.slice(i+1, j)));
                    return tokens;
                }
            }

            if (operatorAndPriority == 4 /* unary ! */) {
                // Special case: the unary operator ! needs to evaluate right-to-left.
                i = tokens.lastIndexOf('!');
                var innerExpr = buildExprTree(tokens.slice(i+1, i+2));
                tokens.splice(i, 2, function() { return !innerExpr(); })
                return tokens;
            }
        })
    }
}

```

```

// A binary operator:
if (operatorAndPriority >= 0) {
    var left = buildExprTree(tokens.slice(0, i));
    var right = buildExprTree(tokens.slice(i+1));
    switch(tokens[i]) {
        case '&&': return [function() { return left() && right(); }];
        case '||': return [function() { return left() || right(); }];
        case '==': return [function() { return left() == right(); }];
        case '!=': return [function() { return left() != right(); }];
        case '<': return [function() { return left() < right(); }];
        case '<=': return [function() { return left() <= right(); }];
        case '>': return [function() { return left() > right(); }];
        case '>=': return [function() { return left() >= right(); }];
        case '+': return [function() { return left() + right(); }];
        case '-': return [function() { return left() - right(); }];
        case '*': return [function() { return left() * right(); }];
        case '/': return [function() { return Math.floor(left() / right()); }];
    }];
}
}

// else a number:
if (tokens[i] === ')') throw 'Parsing failure, mismatched parentheses in
parsing!' + tokens.toString();
assert(operatorAndPriority == -1);
var num = stoi_q(tokens[i]);
return [function() { return num; }]
})(tokens);
}
return tokens[0];
}

// Preprocess the input one line at a time.
for(; i < len; ++i) {
    // Find the start of the current line.
    var lineStart = i;

    // Seek iterator to end of current line.
    i = code.indexOf('\n', i);
    if (i < 0) i = len;

    // Find the first non-whitespace character on the line.
    for(var j = lineStart; j < i && isWhitespace(code, j); ++j);

    // Is this a non-preprocessor directive line?
    var thisLineIsInActivePreprocessingBlock = stack[stack.length-1];
    if (code[j] != '#') { // non-preprocessor line?
        if (thisLineIsInActivePreprocessingBlock) {
            out += expandMacros(code, lineStart, i) + '\n';
        }
        continue;
    }
    // This is a preprocessor directive line, e.g. #ifdef or #define.

    // Parse the line as #<directive> <expression>
    var space = nextWhitespace(code, j);
}

```

```

var directive = code.substring(j+1, space);
var expression = code.substring(space, i).trim();
switch(directive) {
case 'if':
    var tokens = tokenize(expandMacros(expression, 0));
    var exprTree = buildExprTree(tokens);
    var evaluated = exprTree();
    stack.push(!evaluated * stack[stack.length-1]);
    break;
case 'ifdef': stack.push(!defs[expression] * stack[stack.length-1]); break;
case 'ifndef': stack.push(!defs[expression] * stack[stack.length-1]); break;
case 'else': stack[stack.length-1] = 1-stack[stack.length-1]; break;
case 'endif': stack.pop(); break;
case 'define':
    if (thisLineIsInActivePreprocessingBlock) {
        // This could either be a macro with input args (#define MACRO(x,y) x+y),
        or a direct expansion #define FOO 2,
        // figure out which.
        var macroStart = expression.indexOf('(');
        var firstWs = nextWhitespace(expression, 0);
        if (firstWs < macroStart) macroStart = 0;
        if (macroStart > 0) { // #define MACRO( x , y , z ) <statement of x,y and
z>
            var macroEnd = expression.indexOf(')', macroStart);
            let params = expression.substring(macroStart+1,
macroEnd).split(',').map(x => x.trim());
            let value = tokenize(expression.substring(macroEnd+1).trim())
            defs[expression.substring(0, macroStart)] = (args) => {
                var ret = '';
                value.forEach((x) => {
                    var argIndex = params.indexOf(x);
                    ret += (argIndex >= 0) ? args[argIndex] : x;
                });
                return ret;
            };
            } else { // #define FOO (x + y + z)
            let value = expandMacros(expression.substring(firstWs+1).trim(), 0);
            defs[expression.substring(0, firstWs)] = () => value;
        }
    }
    break;
case 'undef': if (thisLineIsInActivePreprocessingBlock) delete
defs[expression]; break;
default:
    if (directive != 'version' && directive != 'pragma' && directive !=
'extension') { // GLSL shader compiler specific #directives.
        err('Unrecognized preprocessor directive #' + directive + '!');
    }

    // Unknown preprocessor macro, just pass through the line to output.
    out += expandMacros(code, lineStart, i) + '\n';
}
}
return out;
}

```

```

function remove_cpp_comments_in_shaders(code) {
    var i = 0, out = '', ch, next, len = code.length;
    for(; i < len; ++i) {
        ch = code[i];
        if (ch == '/') {
            next = code[i+1];
            if (next == '/') {
                while(i < len && code[i+1] != '\n') ++i;
            } else if (next == '*') {
                while(i < len && (code[i-1] != '*' || code[i] != '/')) ++i;
            } else {
                out += ch;
            }
        } else {
            out += ch;
        }
    }
    return out;
}

function _glShaderSource(shader, count, string, length) {
    var source = GL.getSource(shader, count, string, length);

    // These are not expected to be meaningful in WebGL, but issue a warning if they
    // are present, to give some diagnostics about if they are present.
    if (source.includes('__FILE__')) warnOnce('When compiling shader: ' + source + ':'
Preprocessor variable __FILE__ is not handled by -sGL_EXPLICIT_UNIFORM_LOCATION/-sGL_EXPLICIT_UNIFORM_BINDING options!');
    if (source.includes('__LINE__')) warnOnce('When compiling shader: ' + source + ':'
Preprocessor variable __LINE__ is not handled by -sGL_EXPLICIT_UNIFORM_LOCATION/-sGL_EXPLICIT_UNIFORM_BINDING options!');
    // Remove comments and C-preprocess the input shader first, so that we can
    // appropriately
    // parse the layout location directives.
    source = preprocess_c_code(remove_cpp_comments_in_shaders(source), {
        'GL_FRAGMENT_PRECISION_HIGH': () => 1,
        'GL_ES': () => 1,
        '__VERSION__': () => source.includes('#version 300') ? 300 : 100
    });

    // Extract the layout(location = x) directives.
    var regex = /layout\s*\(\s*location\s*=\s*(\d+)\)\s*\((uniform\s+(lowp|mediump|highp)\s+)?\w+\s+(\w+)\)/g,
    explicitUniformLocations = {}, match;
    while(match = regex.exec(source)) {
        explicitUniformLocations[match[5]] = parseInt(match[1]);
        if (!(explicitUniformLocations[match[5]] >= 0 &&
explicitUniformLocations[match[5]] < 1048576)) {
            err('Specified an out of range layout(location=x) directive "' +
explicitUniformLocations[match[5]] + '"! (' + match[0] + ')');
            GL.recordError(0x501 /* GL_INVALID_VALUE */);
            return;
        }
    }

    // Remove all the layout(location = x) directives so that they do not make

```

```

// their way to the actual WebGL shader compiler.
source = source.replace(regex, '$2');

// Remember all the directives to be handled after glLinkProgram is called.
GL.shaders[shader].explicitUniformLocations = explicitUniformLocations;

// Extract the layout(binding = x) directives. Four types we need to handle:
// layout(binding = 3) uniform sampler2D mainTexture;
// layout(binding = 1, std140) uniform MainBlock { ... };
// layout(std140, binding = 1) uniform MainBlock { ... };
// layout(binding = 1) uniform MainBlock { ... };
var bindingRegex = /layout\s*\(\.*?binding\s*=\s*(-\d+).*\?\)\s*uniform\s+(\w+)\s+(\w+)?/g, samplerBindings = {}, uniformBindings = {}, bindingMatch;
while(bindingMatch = bindingRegex.exec(source)) {
    // We have a layout(binding=x) enabled uniform. Parse the array length of that
uniform, if it is an array, i.e. a
    // layout(binding = 3) uniform sampler2D mainTexture[arrayLength];
    // or
    // layout(binding = 1, std140) uniform MainBlock { ... } name[arrayLength];
var arrayLength = 1;
for(var i = bindingMatch.index; i < source.length && source[i] != ';' ; ++i) {
    if (source[i] == '[') {
        arrayLength = jstoi_q(source.slice(i+1));
        break;
    }
    if (source[i] == '{') i = find_closing_parens_index(source, i, '{', '}') - 1;
}
var binding = jstoi_q(bindingMatch[1]);
var bindingsType = 0x8872/*GL_MAX_TEXTURE_IMAGE_UNITS*/;
if (bindingMatch[3] && bindingMatch[2].indexOf('sampler') != -1) {
    samplerBindings[bindingMatch[3]] = [binding, arrayLength];
} else {
    bindingsType = 0x8A2E/*GL_MAX_COMBINED_UNIFORM_BLOCKS*/;
    uniformBindings[bindingMatch[2]] = [binding, arrayLength];
}
var numBindingPoints = GLctx.getParameter(bindingsType);
if (!(binding >= 0 && binding + arrayLength <= numBindingPoints)) {
    err('Specified an out of range layout(binding=x) directive "' + binding + '"!
(' + bindingMatch[0] + '). Valid range is [0, ' + numBindingPoints + '-1]');
    GL.recordError(0x501 /* GL_INVALID_VALUE */);
    return;
}
}

// Remove all the layout(binding = x) directives so that they do not make
// their way to the actual WebGL shader compiler. These regexes get quite hairy,
check against
// https://regex101.com/ when working on these.
source = source.replace(/layout\s*\(\.*?binding\s*=\s*([-d]+)\.*?\)/g, ''); // "
"layout(binding = 3)" -> ""
source = source.replace(/(layout\s*\((.*?)),\s*binding\s*=\s*([-d]+)\)/g,
'$1'); // "layout(std140, binding = 1)" -> "layout(std140)"
source = source.replace(/layout\s*\((\s*binding\s*=\s*([-d]+)\s*,\s*(.*?))\)/g,
'layout($2)'); // "layout(binding = 1, std140)" -> "layout(std140)"

```

```

// Remember all the directives to be handled after glLinkProgram is called.
GL.shaders[shader].explicitSamplerBindings = samplerBindings;
GL.shaders[shader].explicitUniformBindings = uniformBindings;

    GLctx.shaderSource(GL.shaders[shader], source);
}

function _glStencilFuncSeparate(x0, x1, x2, x3) { GLctx['stencilFuncSeparate'](x0,
x1, x2, x3) }

function _glStencilMask(x0) { GLctx['stencilMask'](x0) }

function _glStencilOpSeparate(x0, x1, x2, x3) { GLctx['stencilOpSeparate'](x0, x1,
x2, x3) }

function _glTexImage2D(target, level, internalFormat, width, height, border, format,
type, pixels) {
    if (GL.currentContext.version >= 2) {
        // WebGL 2 provides new garbage-free entry points to call to WebGL. Use those
always when possible.
        if (GLctx.currentPixelUnpackBufferBinding) {
            GLctx.texImage2D(target, level, internalFormat, width, height, border,
format, type, pixels);
        } else if (pixels) {
            var heap = heapObjectForWebGLType(type);
            GLctx.texImage2D(target, level, internalFormat, width, height, border,
format, type, heap, pixels >> heapAccessShiftForWebGLHeap(heap));
        } else {
            GLctx.texImage2D(target, level, internalFormat, width, height, border,
format, type, null);
        }
        return;
    }
    GLctx.texImage2D(target, level, internalFormat, width, height, border, format,
type, pixels ? emscriptenWebGLGetTexPixelData(type, format, width, height, pixels,
internalFormat) : null);
}

function _glTexImage3D(target, level, internalFormat, width, height, depth, border,
format, type, pixels) {
    if (GLctx.currentPixelUnpackBufferBinding) {
        GLctx['texImage3D'](target, level, internalFormat, width, height, depth,
border, format, type, pixels);
    } else if (pixels) {
        var heap = heapObjectForWebGLType(type);
        GLctx['texImage3D'](target, level, internalFormat, width, height, depth,
border, format, type, heap, pixels >> heapAccessShiftForWebGLHeap(heap));
    } else {
        GLctx['texImage3D'](target, level, internalFormat, width, height, depth,
border, format, type, null);
    }
}

function _glTexParameterf(x0, x1, x2) { GLctx['texParameterf'](x0, x1, x2) }

```

```

function _glTexParameterI(x0, x1, x2) { GLctx['texParameteri'](x0, x1, x2) }

function _glTexParameteriv(target, pname, params) {
    var param = HEAP32[((params)>>2)];
    GLctx.texParameteri(target, pname, param);
}

function _glTexStorage2D(x0, x1, x2, x3, x4) { GLctx['texStorage2D'](x0, x1, x2, x3, x4) }

function _glTexStorage3D(x0, x1, x2, x3, x4, x5) { GLctx['texStorage3D'](x0, x1, x2, x3, x4, x5) }

function _glTexSubImage2D(target, level, xoffset, yoffset, width, height, format,
type, pixels) {
    if (GL.currentContext.version >= 2) {
        // WebGL 2 provides new garbage-free entry points to call to WebGL. Use those
        always when possible.
        if (GLctx.currentPixelUnpackBufferBinding) {
            GLctx.texSubImage2D(target, level, xoffset, yoffset, width, height, format,
type, pixels);
        } else if (pixels) {
            var heap = heapObjectForWebGLType(type);
            GLctx.texSubImage2D(target, level, xoffset, yoffset, width, height, format,
type, heap, pixels >> heapAccessShiftForWebGLHeap(heap));
        } else {
            GLctx.texSubImage2D(target, level, xoffset, yoffset, width, height, format,
type, null);
        }
        return;
    }
    var pixelData = null;
    if (pixels) pixelData = emscriptenWebGLGetTexPixelData(type, format, width,
height, pixels, 0);
    GLctx.texSubImage2D(target, level, xoffset, yoffset, width, height, format, type,
pixelData);
}

function _glTexSubImage3D(target, level, xoffset, yoffset, zoffset, width, height,
depth, format, type, pixels) {
    if (GLctx.currentPixelUnpackBufferBinding) {
        GLctx['texSubImage3D'](target, level, xoffset, yoffset, zoffset, width, height,
depth, format, type, pixels);
    } else if (pixels) {
        var heap = heapObjectForWebGLType(type);
        GLctx['texSubImage3D'](target, level, xoffset, yoffset, zoffset, width, height,
depth, format, type, heap, pixels >> heapAccessShiftForWebGLHeap(heap));
    } else {
        GLctx['texSubImage3D'](target, level, xoffset, yoffset, zoffset, width, height,
depth, format, type, null);
    }
}

var miniTempWebGLFloatBuffers = [];
function _glUniform1fv(location, count, value) {

```

```

    if (GL.currentContext.version >= 2) { // WebGL 2 provides new garbage-free entry
points to call to WebGL. Use those always when possible.
        GLctx.uniform1fv(webglGetUniformLocation(location), HEAPF32, value>>2, count);
        return;
    }

    if (count <= 288) {
        // avoid allocation when uploading few enough uniforms
        var view = miniTempWebGLFloatBuffers[count-1];
        for (var i = 0; i < count; ++i) {
            view[i] = HEAPF32[((value)+(4*i))>>2];
        }
    } else
    {
        var view = HEAPF32.subarray((value)>>2, (value+count*4)>>2);
    }
    GLctx.uniform1fv(webglGetUniformLocation(location), view);
}

function _glUniform1i(location, v0) {
    GLctx.uniform1i(webglGetUniformLocation(location), v0);
}

var __miniTempWebGLIntBuffers = [];
function _glUniform1iv(location, count, value) {

    if (GL.currentContext.version >= 2) { // WebGL 2 provides new garbage-free entry
points to call to WebGL. Use those always when possible.
        GLctx.uniform1iv(webglGetUniformLocation(location), HEAP32, value>>2, count);
        return;
    }

    if (count <= 288) {
        // avoid allocation when uploading few enough uniforms
        var view = __miniTempWebGLIntBuffers[count-1];
        for (var i = 0; i < count; ++i) {
            view[i] = HEAP32[((value)+(4*i))>>2];
        }
    } else
    {
        var view = HEAP32.subarray((value)>>2, (value+count*4)>>2);
    }
    GLctx.uniform1iv(webglGetUniformLocation(location), view);
}

function _glUniform1uiv(location, count, value) {
    GLctx.uniform1uiv(webglGetUniformLocation(location), HEAPU32, value>>2, count);
}

function _glUniform2fv(location, count, value) {

    if (GL.currentContext.version >= 2) { // WebGL 2 provides new garbage-free entry
points to call to WebGL. Use those always when possible.

```

```

    GLctx.uniform2fv(webglGetUniformLocation(location), HEAPF32, value>>2,
count*2);
    return;
}

if (count <= 144) {
// avoid allocation when uploading few enough uniforms
var view = miniTempWebGLFloatBuffers[2*count-1];
for (var i = 0; i < 2*count; i += 2) {
    view[i] = HEAPF32[((value)+(4*i))>>2]];
    view[i+1] = HEAPF32[((value)+(4*i+4))>>2]];
}
} else
{
    var view = HEAPF32.subarray((value)>>2, (value+count*8)>>2);
}
GLctx.uniform2fv(webglGetUniformLocation(location), view);
}

function _glUniform2iv(location, count, value) {

if (GL.currentContext.version >= 2) { // WebGL 2 provides new garbage-free entry
points to call to WebGL. Use those always when possible.
    GLctx.uniform2iv(webglGetUniformLocation(location), HEAP32, value>>2, count*2);
    return;
}

if (count <= 144) {
// avoid allocation when uploading few enough uniforms
var view = __miniTempWebGLIntBuffers[2*count-1];
for (var i = 0; i < 2*count; i += 2) {
    view[i] = HEAP32[((value)+(4*i))>>2]];
    view[i+1] = HEAP32[((value)+(4*i+4))>>2]];
}
} else
{
    var view = HEAP32.subarray((value)>>2, (value+count*8)>>2);
}
GLctx.uniform2iv(webglGetUniformLocation(location), view);
}

function _glUniform2uiv(location, count, value) {
    GLctx.uniform2uiv(webglGetUniformLocation(location), HEAPU32, value>>2, count*2);
}

function _glUniform3fv(location, count, value) {

if (GL.currentContext.version >= 2) { // WebGL 2 provides new garbage-free entry
points to call to WebGL. Use those always when possible.
    GLctx.uniform3fv(webglGetUniformLocation(location), HEAPF32, value>>2,
count*3);
    return;
}

if (count <= 96) {
// avoid allocation when uploading few enough uniforms
}
}

```

```

var view = miniTempWebGLFloatBuffers[3*count-1];
for (var i = 0; i < 3*count; i += 3) {
    view[i] = HEAPF32[((value)+(4*i))>>2];
    view[i+1] = HEAPF32[((value)+(4*i+4))>>2];
    view[i+2] = HEAPF32[((value)+(4*i+8))>>2];
}
} else
{
    var view = HEAPF32.subarray((value)>>2, (value+count*12)>>2);
}
GLctx.uniform3fv(webglGetUniformLocation(location), view);
}

function _glUniform3iv(location, count, value) {

    if (GL.currentContext.version >= 2) { // WebGL 2 provides new garbage-free entry
points to call to WebGL. Use those always when possible.
        GLctx.uniform3iv(webglGetUniformLocation(location), HEAP32, value>>2, count*3);
        return;
    }

    if (count <= 96) {
        // avoid allocation when uploading few enough uniforms
        var view = __miniTempWebGLIntBuffers[3*count-1];
        for (var i = 0; i < 3*count; i += 3) {
            view[i] = HEAP32[((value)+(4*i))>>2];
            view[i+1] = HEAP32[((value)+(4*i+4))>>2];
            view[i+2] = HEAP32[((value)+(4*i+8))>>2];
        }
    } else
    {
        var view = HEAP32.subarray((value)>>2, (value+count*12)>>2);
    }
    GLctx.uniform3iv(webglGetUniformLocation(location), view);
}

function _glUniform3uiv(location, count, value) {
    GLctx.uniform3uiv(webglGetUniformLocation(location), HEAPU32, value>>2, count*3);
}

function _glUniform4fv(location, count, value) {

    if (GL.currentContext.version >= 2) { // WebGL 2 provides new garbage-free entry
points to call to WebGL. Use those always when possible.
        GLctx.uniform4fv(webglGetUniformLocation(location), HEAPF32, value>>2,
count*4);
        return;
    }

    if (count <= 72) {
        // avoid allocation when uploading few enough uniforms
        var view = miniTempWebGLFloatBuffers[4*count-1];
        // hoist the heap out of the loop for size and for pthreads+growth.
        var heap = HEAPF32;
        value >>= 2;

```

```

        for (var i = 0; i < 4 * count; i += 4) {
            var dst = value + i;
            view[i] = heap[dst];
            view[i + 1] = heap[dst + 1];
            view[i + 2] = heap[dst + 2];
            view[i + 3] = heap[dst + 3];
        }
    } else
    {
        var view = HEAPF32.subarray((value)>>2, (value+count*16)>>2);
    }
    GLctx.uniform4fv(webglGetUniformLocation(location), view);
}

function _glUniform4iv(location, count, value) {

    if (GL.currentContext.version >= 2) { // WebGL 2 provides new garbage-free entry
points to call to WebGL. Use those always when possible.
        GLctx.uniform4iv(webglGetUniformLocation(location), HEAP32, value>>2, count*4);
        return;
    }

    if (count <= 72) {
        // avoid allocation when uploading few enough uniforms
        var view = __miniTempWebGLIntBuffers[4*count-1];
        for (var i = 0; i < 4*count; i += 4) {
            view[i] = HEAP32[((value)+(4*i))>>2];
            view[i+1] = HEAP32[((value)+(4*i+4))>>2];
            view[i+2] = HEAP32[((value)+(4*i+8))>>2];
            view[i+3] = HEAP32[((value)+(4*i+12))>>2];
        }
    } else
    {
        var view = HEAP32.subarray((value)>>2, (value+count*16)>>2);
    }
    GLctx.uniform4iv(webglGetUniformLocation(location), view);
}

function _glUniform4uiv(location, count, value) {
    GLctx.uniform4uiv(webglGetUniformLocation(location), HEAPU32, value>>2, count*4);
}

function _glUniformBlockBinding(program, uniformBlockIndex, uniformBlockBinding) {
    program = GL.programs[program];

    GLctx['uniformBlockBinding'](program, uniformBlockIndex, uniformBlockBinding);
}

function _glUniformMatrix3fv(location, count, transpose, value) {

    if (GL.currentContext.version >= 2) { // WebGL 2 provides new garbage-free entry
points to call to WebGL. Use those always when possible.
        GLctx.uniformMatrix3fv(webglGetUniformLocation(location), !!transpose, HEAPF32,
value>>2, count*9);
        return;
}

```

```

}

if (count <= 32) {
    // avoid allocation when uploading few enough uniforms
    var view = miniTempWebGLFloatBuffers[9*count-1];
    for (var i = 0; i < 9*count; i += 9) {
        view[i] = HEAPF32[((value)+(4*i))>>2];
        view[i+1] = HEAPF32[((value)+(4*i+4))>>2];
        view[i+2] = HEAPF32[((value)+(4*i+8))>>2];
        view[i+3] = HEAPF32[((value)+(4*i+12))>>2];
        view[i+4] = HEAPF32[((value)+(4*i+16))>>2];
        view[i+5] = HEAPF32[((value)+(4*i+20))>>2];
        view[i+6] = HEAPF32[((value)+(4*i+24))>>2];
        view[i+7] = HEAPF32[((value)+(4*i+28))>>2];
        view[i+8] = HEAPF32[((value)+(4*i+32))>>2];
    }
} else
{
    var view = HEAPF32.subarray((value)>>2, (value+count*36)>>2);
}
GLctx.uniformMatrix3fv(webglGetUniformLocation(location), !transpose, view);
}

function _glUniformMatrix4fv(location, count, transpose, value) {

    if (GL.currentContext.version >= 2) { // WebGL 2 provides new garbage-free entry
points to call to WebGL. Use those always when possible.
        GLctx.uniformMatrix4fv(webglGetUniformLocation(location), !transpose, HEAPF32,
value>>2, count*16);
        return;
    }

    if (count <= 18) {
        // avoid allocation when uploading few enough uniforms
        var view = miniTempWebGLFloatBuffers[16*count-1];
        // hoist the heap out of the loop for size and for pthreads+growth.
        var heap = HEAPF32;
        value >= 2;
        for (var i = 0; i < 16 * count; i += 16) {
            var dst = value + i;
            view[i] = heap[dst];
            view[i + 1] = heap[dst + 1];
            view[i + 2] = heap[dst + 2];
            view[i + 3] = heap[dst + 3];
            view[i + 4] = heap[dst + 4];
            view[i + 5] = heap[dst + 5];
            view[i + 6] = heap[dst + 6];
            view[i + 7] = heap[dst + 7];
            view[i + 8] = heap[dst + 8];
            view[i + 9] = heap[dst + 9];
            view[i + 10] = heap[dst + 10];
            view[i + 11] = heap[dst + 11];
            view[i + 12] = heap[dst + 12];
            view[i + 13] = heap[dst + 13];
            view[i + 14] = heap[dst + 14];
        }
    }
}

```

```

        view[i + 15] = heap[dst + 15];
    }
} else {
{
    var view = HEAPF32.subarray((value)>>2, (value+count*64)>>2);
}
GLctx.uniformMatrix4fv(webglGetUniformLocation(location), !transpose, view);
}

function _glUnmapBuffer(target) {
if (!emscriptenWebGLValidateMapBufferTarget(target)) {
    GL.recordError(0x500/*GL_INVALID_ENUM*/);
    err('GL_INVALID_ENUM in glUnmapBuffer');
    return 0;
}

var buffer = emscriptenWebGLGetBufferBinding(target);
var mapping = GL.mappedBuffers[buffer];
if (!mapping) {
    GL.recordError(0x502 /* GL_INVALID_OPERATION */);
    err('buffer was never mapped in glUnmapBuffer');
    return 0;
}
GL.mappedBuffers[buffer] = null;

if (!(mapping.access & 0x10)) /* GL_MAP_FLUSH_EXPLICIT_BIT */
    if (GL.currentContext.version >= 2) { // WebGL 2 provides new garbage-free
entry points to call to WebGL. Use those always when possible.
        GLctx.bufferSubData(target, mapping.offset, HEAPU8, mapping.mem,
mapping.length);
    } else {
        GLctx.bufferSubData(target, mapping.offset, HEAPU8.subarray(mapping.mem,
mapping.mem+mapping.length));
    }
    _free(mapping.mem);
    return 1;
}

function webglApplyExplicitProgramBindings() {
var p = GLctx.currentProgram;
if (!p.explicitProgramBindingsApplied) {
    if (GL.currentContext.version >= 2) {
        Object.keys(p.explicitUniformBindings).forEach(function(ubo) {
            var bindings = p.explicitUniformBindings[ubo];
            for(var i = 0; i < bindings[1]; ++i) {
                var blockIndex = GLctx.getUniformBlockIndex(p, ubo + (bindings[1] > 1 ?
'[' + i + ']' : ''));
                GLctx.uniformBlockBinding(p, blockIndex, bindings[0]+i);
            }
        });
    }
    Object.keys(p.explicitSamplerBindings).forEach(function(sampler) {
        var bindings = p.explicitSamplerBindings[sampler];
        for(var i = 0; i < bindings[1]; ++i) {

```

```

        GLctx.uniform1i(GLctx.getUniformLocation(p, sampler + (i ? '['+i+']' : '')), bindings[0]+i);
    }
}
p.explicitProgramBindingsApplied = 1;
}
}

function _glUseProgram(program) {
program = GL.programs[program];
GLctx.useProgram(program);
// Record the currently active program so that we can access the uniform
// mapping table of that program.
if ((GLctx.currentProgram = program)) {
    webglApplyExplicitProgramBindings();
}
}

function _glValidateProgram(program) {
    GLctx.validateProgram(GL.programs[program]);
}

function _glVertexAttrib4f(x0, x1, x2, x3, x4) { GLctx['vertexAttrib4f'](x0, x1, x2,
x3, x4) }

function _glVertexAttrib4fv(index, v) {

    GLctx.vertexAttrib4f(index, HEAPF32[v>>2], HEAPF32[v+4>>2], HEAPF32[v+8>>2],
HEAPF32[v+12>>2]);
}

function _glVertexAttribIPointer(index, size, type, stride, ptr) {
    var cb = GL.currentThread.clientBuffers[index];
    if (!GLctx.currentArrayBufferBinding) {
        cb.size = size;
        cb.type = type;
        cb.normalized = false;
        cb.stride = stride;
        cb.ptr = ptr;
        cb.clientside = true;
        cb.vertexAttribPointerAdaptor = function(index, size, type, normalized, stride,
ptr) {
            this.vertexAttribIPointer(index, size, type, stride, ptr);
        };
        return;
    }
    cb.clientside = false;
    GLctx['vertexAttribIPointer'](index, size, type, stride, ptr);
}

function _glVertexAttribPointer(index, size, type, normalized, stride, ptr) {
    var cb = GL.currentThread.clientBuffers[index];
    if (!GLctx.currentArrayBufferBinding) {
        cb.size = size;
        cb.type = type;
        cb.normalized = normalized;
        cb.stride = stride;
    }
}

```

```

cb.ptr = ptr;
cb.clientside = true;
cb.vertexAttribPointerAdaptor = function(index, size, type, normalized, stride,
ptr) {
    this.vertexAttribPointer(index, size, type, normalized, stride, ptr);
};

function _glViewport(x0, x1, x2, x3) { GLctx['viewport'](x0, x1, x2, x3) }

function _llvm_eh_typeid_for(type) {
    return type;
}

function _setTempRet0(val) {
    setTempRet0(val);
}

function __isLeapYear(year) {
    return year%4 === 0 && (year%100 !== 0 || year%400 === 0);
}

function __arraySum(array, index) {
    var sum = 0;
    for (var i = 0; i <= index; sum += array[i++]) {
        // no-op
    }
    return sum;
}

var __MONTH_DAYS_LEAP = [31,29,31,30,31,30,31,31,30,31,30,31];
var __MONTH_DAYS_REGULAR = [31,28,31,30,31,30,31,31,30,31,30,31];
function __addDays(date, days) {
    var newDate = new Date(date.getTime());
    while (days > 0) {
        var leap = __isLeapYear(newDate.getFullYear());
        var currentMonth = newDate.getMonth();
        var daysInCurrentMonth = (leap ? __MONTH_DAYS_LEAP :
__MONTH_DAYS_REGULAR)[currentMonth];

        if (days > daysInCurrentMonth-newDate.getDate()) {
            // we spill over to next month
            days -= (daysInCurrentMonth-newDate.getDate()+1);
            newDate.setDate(1);
            if (currentMonth < 11) {
                newDate.setMonth(currentMonth+1)
            } else {
                newDate.setMonth(0);
                newDate.setFullYear(newDate.getFullYear()+1);
            }
        }
    }
}

```

```

        } else {
            // we stay in current month
            newDate.setDate(newDate.getDate() + days);
            return newDate;
        }
    }

    return newDate;
}
function _strftime(s, maxsize, format, tm) {
    // size_t strftime(char *restrict s, size_t maxsize, const char *restrict format,
    const struct tm *restrict timeptr);
    // http://pubs.opengroup.org/onlinepubs/009695399/functions/strftime.html

var tm_zone = HEAP32[((tm)+(40))>>2];

var date = {
    tm_sec: HEAP32[((tm)>>2)],
    tm_min: HEAP32[((tm)+(4))>>2]],
    tm_hour: HEAP32[((tm)+(8))>>2]],
    tm_mday: HEAP32[((tm)+(12))>>2]],
    tm_mon: HEAP32[((tm)+(16))>>2]],
    tm_year: HEAP32[((tm)+(20))>>2]],
    tm_wday: HEAP32[((tm)+(24))>>2]],
    tm_yday: HEAP32[((tm)+(28))>>2]],
    tm_isdst: HEAP32[((tm)+(32))>>2]],
    tm_gmtoff: HEAP32[((tm)+(36))>>2]],
    tm_zone: tm_zone ? UTF8ToString(tm_zone) : ''
};

var pattern = UTF8ToString(format);

// expand format
var EXPANSION_RULES_1 = {
    '%c': '%a %b %d %H:%M:%S %Y',      // Replaced by the locale's appropriate date
and time representation - e.g., Mon Aug 3 14:02:01 2013
    '%D': '%m/%d/%y',                  // Equivalent to %m / %d / %y
    '%F': '%Y-%m-%d',                 // Equivalent to %Y - %m - %d
    '%h': '%b',                      // Equivalent to %b
    '%r': '%I:%M:%S %p',              // Replaced by the time in a.m. and p.m.
notation
    '%R': '%H:%M',                   // Replaced by the time in 24-hour notation
    '%T': '%H:%M:%S',                // Replaced by the time
    '%X': '%m/%d/%y',                // Replaced by the locale's appropriate date
representation
    '%x': '%H:%M:%S',                // Replaced by the locale's appropriate time
representation
    // Modified Conversion Specifiers
    '%Ec': '%c',                     // Replaced by the locale's alternative
appropriate date and time representation.
    '%EC': '%C',                     // Replaced by the name of the base year
(period) in the locale's alternative representation.
    '%Ex': '%m/%d/%y',                // Replaced by the locale's alternative date
representation.
}

```

```

        '%EX': '%H:%M:%S',                                // Replaced by the locale's alternative time
representation.
        '%Ey': '%y',                                     // Replaced by the offset from %EC (year
only) in the locale's alternative representation.
        '%EY': '%Y',                                     // Replaced by the full alternative year
representation.
        '%Od': '%d',                                     // Replaced by the day of the month, using
the locale's alternative numeric symbols, filled as needed with leading zeros if there
is any alternative symbol for zero; otherwise, with leading <space> characters.
        '%Oe': '%e',                                     // Replaced by the day of the month, using
the locale's alternative numeric symbols, filled as needed with leading <space>
characters.
        '%OH': '%H',                                     // Replaced by the hour (24-hour clock) using
the locale's alternative numeric symbols.
        '%OI': '%I',                                     // Replaced by the hour (12-hour clock) using
the locale's alternative numeric symbols.
        '%Om': '%m',                                     // Replaced by the month using the locale's
alternative numeric symbols.
        '%OM': '%M',                                     // Replaced by the minutes using the locale's
alternative numeric symbols.
        '%OS': '%S',                                     // Replaced by the seconds using the locale's
alternative numeric symbols.
        '%Ou': '%u',                                     // Replaced by the weekday as a number in the
locale's alternative representation (Monday=1).
        '%OU': '%U',                                     // Replaced by the week number of the year
(Sunday as the first day of the week, rules corresponding to %U ) using the locale's
alternative numeric symbols.
        '%OV': '%V',                                     // Replaced by the week number of the year
(Monday as the first day of the week, rules corresponding to %V ) using the locale's
alternative numeric symbols.
        '%Ow': '%w',                                     // Replaced by the number of the weekday
(Sunday=0) using the locale's alternative numeric symbols.
        '%OW': '%W',                                     // Replaced by the week number of the year
(Monday as the first day of the week) using the locale's alternative numeric symbols.
        '%Oy': '%y',                                     // Replaced by the year (offset from %C )
using the locale's alternative numeric symbols.
    };
    for (var rule in EXPANSION_RULES_1) {
        pattern = pattern.replace(new RegExp(rule, 'g'), EXPANSION_RULES_1[rule]);
    }

    var WEEKDAYS = ['Sunday', 'Monday', 'Tuesday', 'Wednesday', 'Thursday', 'Friday',
'Saturday'];
    var MONTHS = ['January', 'February', 'March', 'April', 'May', 'June', 'July',
'August', 'September', 'October', 'November', 'December'];

    function leadingSomething(value, digits, character) {
        var str = typeof value == 'number' ? value.toString() : (value || '');
        while (str.length < digits) {
            str = character[0]+str;
        }
        return str;
    }

    function leadingNulls(value, digits) {
        return leadingSomething(value, digits, '0');
    }
}

```

```

}

function compareByDay(date1, date2) {
    function sgn(value) {
        return value < 0 ? -1 : (value > 0 ? 1 : 0);
    }

    var compare;
    if ((compare = sgn(date1.getFullYear() - date2.getFullYear())) === 0) {
        if ((compare = sgn(date1.getMonth() - date2.getMonth())) === 0) {
            compare = sgn(date1.getDate() - date2.getDate());
        }
    }
    return compare;
}

function getFirstWeekStartDate(janFourth) {
    switch (janFourth.getDay()) {
        case 0: // Sunday
            return new Date(janFourth.getFullYear() - 1, 11, 29);
        case 1: // Monday
            return janFourth;
        case 2: // Tuesday
            return new Date(janFourth.getFullYear(), 0, 3);
        case 3: // Wednesday
            return new Date(janFourth.getFullYear(), 0, 2);
        case 4: // Thursday
            return new Date(janFourth.getFullYear(), 0, 1);
        case 5: // Friday
            return new Date(janFourth.getFullYear() - 1, 11, 31);
        case 6: // Saturday
            return new Date(janFourth.getFullYear() - 1, 11, 30);
    }
}

function getWeekBasedYear(date) {
    var thisDate = __addDays(new Date(date.tm_year + 1900, 0, 1), date.tm_yday);

    var janFourthThisYear = new Date(thisDate.getFullYear(), 0, 4);
    var janFourthNextYear = new Date(thisDate.getFullYear() + 1, 0, 4);

    var firstWeekStartThisYear = getFirstWeekStartDate(janFourthThisYear);
    var firstWeekStartNextYear = getFirstWeekStartDate(janFourthNextYear);

    if (compareByDay(firstWeekStartThisYear, thisDate) <= 0) {
        // this date is after the start of the first week of this year
        if (compareByDay(firstWeekStartNextYear, thisDate) <= 0) {
            return thisDate.getFullYear() + 1;
        } else {
            return thisDate.getFullYear();
        }
    } else {
        return thisDate.getFullYear() - 1;
    }
}

```

```

var EXPANSION_RULES_2 = {
    '%a': function(date) {
        return WEEKDAYS[date.tm_wday].substring(0,3);
    },
    '%A': function(date) {
        return WEEKDAYS[date.tm_wday];
    },
    '%b': function(date) {
        return MONTHS[date.tm_mon].substring(0,3);
    },
    '%B': function(date) {
        return MONTHS[date.tm_mon];
    },
    '%C': function(date) {
        var year = date.tm_year+1900;
        return leadingNulls((year/100)|0,2);
    },
    '%d': function(date) {
        return leadingNulls(date.tm_mday, 2);
    },
    '%e': function(date) {
        return leadingSomething(date.tm_mday, 2, ' ');
    },
    '%g': function(date) {
        // %g, %G, and %V give values according to the ISO 8601:2000 standard week-based year.
        // In this system, weeks begin on a Monday and week 1 of the year is the week that includes
        // January 4th, which is also the week that includes the first Thursday of the year, and
        // is also the first week that contains at least four days in the year.
        // If the first Monday of January is the 2nd, 3rd, or 4th, the preceding days are part of
        // the last week of the preceding year; thus, for Saturday 2nd January 1999,
        // %G is replaced by 1998 and %V is replaced by 53. If December 29th, 30th,
        // or 31st is a Monday, it and any following days are part of week 1 of the following year.
        // Thus, for Tuesday 30th December 1997, %G is replaced by 1998 and %V is replaced by 01.
        return getWeekBasedYear(date).toString().substring(2);
    },
    '%G': function(date) {
        return getWeekBasedYear(date);
    },
    '%H': function(date) {
        return leadingNulls(date.tm_hour, 2);
    },
    '%I': function(date) {
        var twelveHour = date.tm_hour;
        if (twelveHour == 0) twelveHour = 12;
        else if (twelveHour > 12) twelveHour -= 12;
        return leadingNulls(twelveHour, 2);
    },
}

```

```

'%j': function(date) {
    // Day of the year (001-366)
    return leadingNulls(date.tm_mday+__arraySum(__isLeapYear(date.tm_year+1900) ?
__MONTH_DAYS_LEAP : __MONTH_DAYS_REGULAR, date.tm_mon-1), 3);
},
'%m': function(date) {
    return leadingNulls(date.tm_mon+1, 2);
},
'%M': function(date) {
    return leadingNulls(date.tm_min, 2);
},
'%n': function() {
    return '\n';
},
'%p': function(date) {
    if (date.tm_hour >= 0 && date.tm_hour < 12) {
        return 'AM';
    } else {
        return 'PM';
    }
},
'%S': function(date) {
    return leadingNulls(date.tm_sec, 2);
},
'%t': function() {
    return '\t';
},
'%u': function(date) {
    return date.tm_wday || 7;
},
'%U': function(date) {
    var days = date.tm_yday + 7 - date.tm_wday;
    return leadingNulls(Math.floor(days / 7), 2);
},
'%V': function(date) {
    // Replaced by the week number of the year (Monday as the first day of the
week)
    // as a decimal number [01,53]. If the week containing 1 January has four
    // or more days in the new year, then it is considered week 1.
    // Otherwise, it is the last week of the previous year, and the next week is
week 1.
    // Both January 4th and the first Thursday of January are always in week 1. [
tm_year, tm_wday, tm_yday]
    var val = Math.floor((date.tm_yday + 7 - (date.tm_wday + 6) % 7) / 7);
    // If 1 Jan is just 1-3 days past Monday, the previous week
    // is also in this year.
    if ((date.tm_wday + 371 - date.tm_yday - 2) % 7 <= 2) {
        val++;
    }
    if (!val) {
        val = 52;
        // If 31 December of prev year a Thursday, or Friday of a
        // leap year, then the prev year has 53 weeks.
        var dec31 = (date.tm_wday + 7 - date.tm_yday - 1) % 7;
        if (dec31 == 4 || (dec31 == 5 && __isLeapYear(date.tm_year%400-1))) {

```

```

        val++;
    }
} else if (val == 53) {
    // If 1 January is not a Thursday, and not a Wednesday of a
    // leap year, then this year has only 52 weeks.
    var jan1 = (date.tm_wday + 371 - date.tm_yday) % 7;
    if (jan1 != 4 && (jan1 != 3 || !__isLeapYear(date.tm_year)))
        val = 1;
}
return leadingNulls(val, 2);
},
'%w': function(date) {
    return date.tm_wday;
},
'%W': function(date) {
    var days = date.tm_yday + 7 - ((date.tm_wday + 6) % 7);
    return leadingNulls(Math.floor(days / 7), 2);
},
'%y': function(date) {
    // Replaced by the last two digits of the year as a decimal number [00,99]. [tm_year]
    return (date.tm_year+1900).toString().substring(2);
},
'%Y': function(date) {
    // Replaced by the year as a decimal number (for example, 1997). [tm_year]
    return date.tm_year+1900;
},
'%z': function(date) {
    // Replaced by the offset from UTC in the ISO 8601:2000 standard format (+hhmm or -hhmm).
    // For example, "-0430" means 4 hours 30 minutes behind UTC (west of Greenwich).
    var off = date.tm_gmtoff;
    var ahead = off >= 0;
    off = Math.abs(off) / 60;
    // convert from minutes into hhmm format (which means 60 minutes = 100 units)
    off = (off / 60)*100 + (off % 60);
    return (ahead ? '+' : '-') + String("0000" + off).slice(-4);
},
'%Z': function(date) {
    return date.tm_zone;
},
'%%': function() {
    return '%';
}
};

// Replace %% with a pair of NULLs (which cannot occur in a C string), then
// re-inject them after processing.
pattern = pattern.replace(/%/g, '\0\0')
for (var rule in EXPANSION_RULES_2) {
    if (pattern.includes(rule)) {
        pattern = pattern.replace(new RegExp(rule, 'g'),
EXPANSION_RULES_2[rule](date));
    }
}
```

```

        }
        pattern = pattern.replace(/\0\0/g, '%')

    var bytes = intArrayFromString(pattern, false);
    if (bytes.length > maxsize) {
        return 0;
    }

    writeArrayToMemory(bytes, s);
    return bytes.length-1;
}

var FSNode = /** @constructor */ function(parent, name, mode, rdev) {
    if (!parent) {
        parent = this; // root node sets parent to itself
    }
    this.parent = parent;
    this.mount = parent.mount;
    this.mounted = null;
    this.id = FS.nextInode++;
    this.name = name;
    this.mode = mode;
    this.node_ops = {};
    this.stream_ops = {};
    this.rdev = rdev;
};

var readMode = 292/*292*/ | 73/*73*/;
var writeMode = 146/*146*/;

Object.defineProperties(FSNode.prototype, {
    read: {
        get: /** @this{FSNode} */function() {
            return (this.mode & readMode) === readMode;
        },
        set: /** @this{FSNode} */function(val) {
            val ? this.mode |= readMode : this.mode &= ~readMode;
        }
    },
    write: {
        get: /** @this{FSNode} */function() {
            return (this.mode & writeMode) === writeMode;
        },
        set: /** @this{FSNode} */function(val) {
            val ? this.mode |= writeMode : this.mode &= ~writeMode;
        }
    },
    isFolder: {
        get: /** @this{FSNode} */function() {
            return FS.isDir(this.mode);
        }
    },
    isDevice: {
        get: /** @this{FSNode} */function() {
            return FS.isChrdev(this.mode);
        }
    }
})

```

```
});
FS.FSNode = FSNode;
FS.staticInit();Module["FS_createPath"] = FS.createPath;Module["FS_createDataFile"] =
FS.createDataFile;
ERRNO_CODES = {
    'EPERM': 63,
    'ENOENT': 44,
    'ESRCH': 71,
    'EINTR': 27,
    'EIO': 29,
    'ENXIO': 60,
    'E2BIG': 1,
    'ENOEXEC': 45,
    'EBADF': 8,
    'ECHILD': 12,
    'EAGAIN': 6,
    'EWOULDBLOCK': 6,
    'ENOMEM': 48,
    'EACCES': 2,
    'EFAULT': 21,
    'ENOTBLK': 105,
    'EBUSY': 10,
    'EEXIST': 20,
    'EXDEV': 75,
    'ENODEV': 43,
    'ENOTDIR': 54,
    'EISDIR': 31,
    'EINVAL': 28,
    'ENFILE': 41,
    'EMFILE': 33,
    'ENOTTY': 59,
    'ETXTBSY': 74,
    'EFBIG': 22,
    'ENOSPC': 51,
    'ESPIPE': 70,
    'EROFS': 69,
    'EMLINK': 34,
    'EPIPE': 64,
    'EDOM': 18,
    'ERANGE': 68,
    'ENOMSG': 49,
    'EIDRM': 24,
    'ECHRNG': 106,
    'EL2NSYNC': 156,
    'EL3HLT': 107,
    'EL3RST': 108,
    'ELNRNG': 109,
    'EUNATCH': 110,
    'ENOCSI': 111,
    'EL2HLT': 112,
    'EDEADLK': 16,
    'ENOLCK': 46,
    'EBADE': 113,
    'EBADR': 114,
    'EXFULL': 115,
```

'ENOANO': 104,
'EBADRQC': 103,
'EBADSLT': 102,
'EDEADLOCK': 16,
'EBFONT': 101,
'ENOSTR': 100,
'ENODATA': 116,
'ETIME': 117,
'ENOSR': 118,
'ENONET': 119,
'ENOPKG': 120,
'EREMOTE': 121,
'ENOLINK': 47,
'EADV': 122,
'ESRMNT': 123,
'ECOMM': 124,
'EPROTO': 65,
'EMULTIHOP': 36,
'EDOTDOT': 125,
'EBADMSG': 9,
'ENOTUNIQ': 126,
'EBADFD': 127,
'EREMCHG': 128,
'ELIBACC': 129,
'ELIBBAD': 130,
'ELIBSCN': 131,
'ELIBMAX': 132,
'ELIBEXEC': 133,
'ENOSYS': 52,
'ENOTEMPTY': 55,
'ENAMETOOLONG': 37,
'ELOOP': 32,
'EOPNOTSUPP': 138,
'EPFNOSUPPORT': 139,
'ECONNRESET': 15,
'ENOBUFS': 42,
'EAFNOSUPPORT': 5,
'EPROTOTYPE': 67,
'ENOTSOCK': 57,
'ENOPROTOOPT': 50,
'ESHUTDOWN': 140,
'ECONNREFUSED': 14,
'EADDRINUSE': 3,
'ECONNABORTED': 13,
'ENETUNREACH': 40,
'ENETDOWN': 38,
'ETIMEDOUT': 73,
'EHOSTDOWN': 142,
'EHOSTUNREACH': 23,
'EINPROGRESS': 26,
'EALREADY': 7,
'EDESTADDRREQ': 17,
'EMSGSIZE': 35,
'EPROTONOSUPPORT': 66,
'ESOCKTNOSUPPORT': 137,

```

'EADDRNOTAVAIL': 4,
'ENETRESET': 39,
'EISCONN': 30,
'ENOTCONN': 53,
'ETOOMANYREFS': 141,
'EUSERS': 136,
'EDQUOT': 19,
'ESTALE': 72,
'ENOTSUP': 138,
'ENOMEDIUM': 148,
'EILSEQ': 25,
'E_OVERFLOW': 61,
'ECANCELED': 11,
'ENOTRECOVERABLE': 56,
'EOWNERDEAD': 62,
'ESTRPIPE': 135,
};

Module["requestFullscreen"] = function Module_requestFullscreen(lockPointer, resizeCanvas) { Browser.requestFullscreen(lockPointer, resizeCanvas) };
Module["requestFullScreen"] = function Module_requestFullScreen() {
Browser.requestFullScreen() };
Module["requestAnimationFrame"] = function Module_requestAnimationFrame(func) {
Browser.requestAnimationFrame(func) };
Module["setCanvasSize"] = function Module_setCanvasSize(width, height, noUpdates) {
Browser.setCanvasSize(width, height, noUpdates) };
Module["pauseMainLoop"] = function Module_pauseMainLoop() { Browser.mainLoop.pause() };
Module["resumeMainLoop"] = function Module_resumeMainLoop() {
Browser.mainLoop.resume() };
Module["getUserMedia"] = function Module_getUserMedia() { Browser.getUserMedia() };
Module["createContext"] = function Module_createContext(canvas, useWebGL, setInModule, webGLContextAttributes) { return Browser.createContext(canvas, useWebGL, setInModule, webGLContextAttributes) };

var GLctx;;
for (var i = 0; i < 32; ++i) tempFixedLengthArray.push(new Array(i));
var miniTempWebGLFloatBuffersStorage = new Float32Array(288);
for (**@suppress{duplicate}**/var i = 0; i < 288; ++i) {
miniTempWebGLFloatBuffers[i] = miniTempWebGLFloatBuffersStorage.subarray(0, i+1);
}
;
var __miniTempWebGLIntBuffersStorage = new Int32Array(288);
for (**@suppress{duplicate}**/var i = 0; i < 288; ++i) {
__miniTempWebGLIntBuffers[i] = __miniTempWebGLIntBuffersStorage.subarray(0, i+1);
}
;
var ASSERTIONS = true;

/** @type {function(string, boolean=, number=)} */
function intArrayFromString(stringy, dontAddNull, length) {
var len = length > 0 ? length : lengthBytesUTF8(stringy)+1;
var u8array = new Array(len);
var numBytesWritten = stringToUTF8Array(stringy, u8array, 0, u8array.length);
if (dontAddNull) u8array.length = numBytesWritten;
}

```

```

        return u8array;
    }

function intArrayToString(array) {
    var ret = [];
    for (var i = 0; i < array.length; i++) {
        var chr = array[i];
        if (chr > 0xFF) {
            if (ASSERTIONS) {
                assert(false, 'Character code ' + chr + ' (' + String.fromCharCode(chr) + ')'
at offset ' + i + ' not in 0x00-0xFF.');
            }
            chr &= 0xFF;
        }
        ret.push(String.fromCharCode(chr));
    }
    return ret.join('');
}

function checkIncomingModuleAPI() {
    ignoredModuleProp('fetchSettings');
}

var asmLibraryArg = {
    "GetJSMemoryInfo": _GetJSMemoryInfo,
    "JS_Accelerometer_IsRunning": _JS_Accelerometer_IsRunning,
    "JS_Accelerometer_Start": _JS_Accelerometer_Start,
    "JS_Accelerometer_Stop": _JS_Accelerometer_Stop,
    "JS_Cursor_SetImage": _JS_Cursor_SetImage,
    "JS_Cursor_SetShow": _JS_Cursor_SetShow,
    "JS_DOM_MapViewportCoordinateToElementLocalCoordinate":
    _JS_DOM_MapViewportCoordinateToElementLocalCoordinate,
    "JS_DOM_UnityCanvasSelector": _JS_DOM_UnityCanvasSelector,
    "JS_Eval_OpenURL": _JS_Eval_OpenURL,
    "JS_FileSystem_Initialize": _JS_FileSystem_Initialize,
    "JS_FileSystem_Sync": _JS_FileSystem_Sync,
    "JS_GravitySensor_IsRunning": _JS_GravitySensor_IsRunning,
    "JS_GravitySensor_Start": _JS_GravitySensor_Start,
    "JS_GravitySensor_Stop": _JS_GravitySensor_Stop,
    "JS_GuardAgainstJsExceptions": _JS_GuardAgainstJsExceptions,
    "JS_Gyroscope_IsRunning": _JS_Gyroscope_IsRunning,
    "JS_Gyroscope_Start": _JS_Gyroscope_Start,
    "JS_Gyroscope_Stop": _JS_Gyroscope_Stop,
    "JS_LinearAccelerationSensor_IsRunning": _JS_LinearAccelerationSensor_IsRunning,
    "JS_LinearAccelerationSensor_Start": _JS_LinearAccelerationSensor_Start,
    "JS_LinearAccelerationSensor_Stop": _JS_LinearAccelerationSensor_Stop,
    "JS_Log_Dump": _JS_Log_Dump,
    "JS_Log_StackTrace": _JS_Log_StackTrace,
    "JS_MobileKeyboard_GetIgnoreBlurEvent": _JS_MobileKeyboard_GetIgnoreBlurEvent,
    "JS_MobileKeyboard_GetKeyboardStatus": _JS_MobileKeyboard_GetKeyboardStatus,
    "JS_MobileKeyboard_GetText": _JS_MobileKeyboard_GetText,
    "JS_MobileKeyboard_GetTextSelection": _JS_MobileKeyboard_GetTextSelection,
    "JS_MobileKeyboard_Hide": _JS_MobileKeyboard_Hide,
    "JS_MobileKeyboard_SetCharacterLimit": _JS_MobileKeyboard_SetCharacterLimit,
    "JS_MobileKeyboard_SetText": _JS_MobileKeyboard_SetText,
}

```

```
"JS_MobileKeyboard_SetTextSelection": _JS_MobileKeyboard_SetTextSelection,
"JS_MobileKeyboard_Show": _JS_MobileKeyboard_Show,
"JS_OrientationSensor_IsRunning": _JS_OrientationSensor_IsRunning,
"JS_OrientationSensor_Start": _JS_OrientationSensor_Start,
"JS_OrientationSensor_Stop": _JS_OrientationSensor_Stop,
"JS_Profiler.InjectJobs": _JS_Profiler.InjectJobs,
"JS_RequestDeviceSensorPermissionsOnTouch":  

-JS_RequestDeviceSensorPermissionsOnTouch,  

"JS_RunQuitCallbacks": _JS_RunQuitCallbacks,
"JS_ScreenOrientation_DeInit": _JS_ScreenOrientation_DeInit,
"JS_ScreenOrientation_Init": _JS_ScreenOrientation_Init,
"JS_ScreenOrientation_Lock": _JS_ScreenOrientation_Lock,
"JS_Sound_ResumeIfNeeded": _JS_Sound_ResumeIfNeeded,
"JS_SystemInfo_GetBrowserName": _JS_SystemInfo_GetBrowserName,
"JS_SystemInfo_GetBrowserVersionString": _JS_SystemInfo_GetBrowserVersionString,
"JS_SystemInfo_GetCanvasClientSize": _JS_SystemInfo_GetCanvasClientSize,
"JS_SystemInfo_GetDocumentURL": _JS_SystemInfo_GetDocumentURL,
"JS_SystemInfo_GetGPUInfo": _JS_SystemInfo_GetGPUInfo,
"JS_SystemInfo_GetLanguage": _JS_SystemInfo_GetLanguage,
"JS_SystemInfo_GetMatchWebGLToCanvasSize": _JS_SystemInfo_GetMatchWebGLToCanvasSize,
"JS_SystemInfo_GetMemory": _JS_SystemInfo_GetMemory,
"JS_SystemInfo_GetOS": _JS_SystemInfo_GetOS,
"JS_SystemInfo_GetPreferredDevicePixelRatio":  

-JS_SystemInfo_GetPreferredDevicePixelRatio,  

"JS_SystemInfo_GetScreenSize": _JS_SystemInfo_GetScreenSize,
"JS_SystemInfo_HasAstcHdr": _JS_SystemInfo_HasAstcHdr,
"JS_SystemInfo_HasCursorLock": _JS_SystemInfo_HasCursorLock,
"JS_SystemInfo_HasFullscreen": _JS_SystemInfo_HasFullscreen,
"JS_SystemInfo_HasWebGL": _JS_SystemInfo_HasWebGL,
"JS_UnityEngineShouldQuit": _JS_UnityEngineShouldQuit,
"JS_WebRequest_Abort": _JS_WebRequest_Abort,
"JS_WebRequest_Create": _JS_WebRequest_Create,
"JS_WebRequest.GetResponseMetaData": _JS_WebRequest.GetResponseMetaData,
"JS_WebRequest.GetResponseMetaDataLengths":  

-JS_WebRequest.GetResponseMetaDataLengths,  

"JS_WebRequest_Release": _JS_WebRequest_Release,
"JS_WebRequest_Send": _JS_WebRequest_Send,
"JS_WebRequest_SetRedirectLimit": _JS_WebRequest_SetRedirectLimit,
"JS_WebRequest_SetRequestHeader": _JS_WebRequest_SetRequestHeader,
"JS_WebRequest_SetTimeout": _JS_WebRequest_SetTimeout,
"__assert_fail": __assert_fail,
"__cxa_allocate_exception": __cxa_allocate_exception,
"__cxa_begin_catch": __cxa_begin_catch,
"__cxa_end_catch": __cxa_end_catch,
"__cxa_find_matching_catch_2": __cxa_find_matching_catch_2,
"__cxa_find_matching_catch_3": __cxa_find_matching_catch_3,
"__cxa_find_matching_catch_4": __cxa_find_matching_catch_4,
"__cxa_free_exception": __cxa_free_exception,
"__cxa_rethrow": __cxa_rethrow,
"__cxa_throw": __cxa_throw,
"__resumeException": __resumeException,
"__syscall_newselect": __syscall_newselect,
"__syscall_accept4": __syscall_accept4,
"__syscall_chmod": __syscall_chmod,
"__syscall_connect": __syscall_connect,
```

```
"__syscall_faccessat": __syscall_faccessat,
"__syscall_fcntl64": __syscall_fcntl64,
"__syscall_fstat64": __syscall_fstat64,
"__syscall_getcwd": __syscall_getcwd,
"__syscall_getdents64": __syscall_getdents64,
"__syscall_getsockopt": __syscall_getsockopt,
"__syscall_ioctl": __syscall_ioctl,
"__syscall_lstat64": __syscall_lstat64,
"__syscall_mkdir": __syscall_mkdir,
"__syscall_newfstatat": __syscall_newfstatat,
"__syscall_openat": __syscall_openat,
"__syscall_readlinkat": __syscall_readlinkat,
"__syscall_recvfrom": __syscall_recvfrom,
"__syscall_renameat": __syscall_renameat,
"__syscall_rmdir": __syscall_rmdir,
"__syscall_sendto": __syscall_sendto,
"__syscall_socket": __syscall_socket,
"__syscall_stat64": __syscall_stat64,
"__syscall_statfs64": __syscall_statfs64,
"__syscall_truncate64": __syscall_truncate64,
"__syscall_unlinkat": __syscall_unlinkat,
"__syscall_utimensat": __syscall_utimensat,
"_dlopen_js": _dlopen_js,
"_dlsym_js": _dlsym_js,
"_emscripten_date_now": _emscripten_date_now,
"_emscripten_get_now_is_monotonic": _emscripten_get_now_is_monotonic,
"_emscripten_throw_longjmp": _emscripten_throw_longjmp,
"_gmtime_js": _gmtime_js,
"_localtime_js": _localtime_js,
"_mktime_js": _mktime_js,
"_mmap_js": _mmap_js,
"_munmap_js": _munmap_js,
"_tzset_js": _tzset_js,
"abort": _abort,
"_emscripten_asm_const_int": _emscripten_asm_const_int,
"_emscripten_asm_const_int_sync_on_main_thread":
_emscripten_asm_const_int_sync_on_main_thread,
"_emscripten_cancel_main_loop": _emscripten_cancel_main_loop,
"_emscripten_clear_interval": _emscripten_clear_interval,
"_emscripten_console_error": _emscripten_console_error,
"_emscripten_exit_fullscreen": _emscripten_exit_fullscreen,
"_emscripten_exit_pointerlock": _emscripten_exit_pointerlock,
"_emscripten_get_canvas_element_size": _emscripten_get_canvas_element_size,
"_emscripten_get_fullscreen_status": _emscripten_get_fullscreen_status,
"_emscripten_get_gamepad_status": _emscripten_get_gamepad_status,
"_emscripten_get_heap_max": _emscripten_get_heap_max,
"_emscripten_get_now": _emscripten_get_now,
"_emscripten_get_now_res": _emscripten_get_now_res,
"_emscripten_get_num_gamepads": _emscripten_get_num_gamepads,
"_emscripten_html5_remove_all_event_listeners":
_emscripten_html5_remove_all_event_listeners,
"_emscripten_is_webgl_context_lost": _emscripten_is_webgl_context_lost,
"_emscripten_log": _emscripten_log,
"_emscripten_memcpy_big": _emscripten_memcpy_big,
"_emscripten_request_fullscreen": _emscripten_request_fullscreen,
```

```
"emscripten_request_pointerlock": _emscripten_request_pointerlock,
"emscripten_resize_heap": _emscripten_resize_heap,
"emscripten_sample_gamepad_data": _emscripten_sample_gamepad_data,
"emscripten_set_blur_callback_on_thread": _emscripten_set_blur_callback_on_thread,
"emscripten_set_canvas_element_size": _emscripten_set_canvas_element_size,
"emscripten_set_focus_callback_on_thread": _emscripten_set_focus_callback_on_thread,
"emscripten_set_fullscreenchange_callback_on_thread":
_emscripten_set_fullscreenchange_callback_on_thread,
"emscripten_set_gamepadconnected_callback_on_thread":
_emscripten_set_gamepadconnected_callback_on_thread,
"emscripten_set_gamepaddisconnected_callback_on_thread":
_emscripten_set_gamepaddisconnected_callback_on_thread,
"emscripten_set_interval": _emscripten_set_interval,
"emscripten_set_keydown_callback_on_thread":
_emscripten_set_keydown_callback_on_thread,
"emscripten_set_keydown_callback_on_thread",
"emscripten_set_keypress_callback_on_thread":
_emscripten_set_keypress_callback_on_thread,
"emscripten_set_keypress_callback_on_thread",
"emscripten_set_keyup_callback_on_thread": _emscripten_set_keyup_callback_on_thread,
"emscripten_set_main_loop": _emscripten_set_main_loop,
"emscripten_set_main_loop_timing": _emscripten_set_main_loop_timing,
"emscripten_set_mousedown_callback_on_thread":
_emscripten_set_mousedown_callback_on_thread,
"emscripten_set_mousedown_callback_on_thread",
"emscripten_set_mousemove_callback_on_thread":
_emscripten_set_mousemove_callback_on_thread,
"emscripten_set_mousemove_callback_on_thread",
"emscripten_set_mouseup_callback_on_thread":
_emscripten_set_mouseup_callback_on_thread,
"emscripten_set_mouseup_callback_on_thread",
"emscripten_set_pointerlockchange_callback_on_thread":
_emscripten_set_pointerlockchange_callback_on_thread,
"emscripten_set_pointerlockchange_callback_on_thread",
"emscripten_set_touchcancel_callback_on_thread":
_emscripten_set_touchcancel_callback_on_thread,
"emscripten_set_touchcancel_callback_on_thread",
"emscripten_set_touchend_callback_on_thread":
_emscripten_set_touchend_callback_on_thread,
"emscripten_set_touchend_callback_on_thread",
"emscripten_set_touchmove_callback_on_thread":
_emscripten_set_touchmove_callback_on_thread,
"emscripten_set_touchmove_callback_on_thread",
"emscripten_set_touchstart_callback_on_thread":
_emscripten_set_touchstart_callback_on_thread,
"emscripten_set_touchstart_callback_on_thread",
"emscripten_set_wheel_callback_on_thread": _emscripten_set_wheel_callback_on_thread,
"emscripten_webgl_create_context": _emscripten_webgl_create_context,
"emscripten_webgl_destroy_context": _emscripten_webgl_destroy_context,
"emscripten_webgl_enable_extension": _emscripten_webgl_enable_extension,
"emscripten_webgl_get_current_context": _emscripten_webgl_get_current_context,
"emscripten_webgl_init_context_attributes":
_emscripten_webgl_init_context_attributes,
"emscripten_webgl_make_context_current": _emscripten_webgl_make_context_current,
"environ_get": _environ_get,
"environ_sizes_get": _environ_sizes_get,
"exit": _exit,
"fd_close": _fd_close,
"fd_fstat_get": _fd_fstat_get,
"fd_read": _fd_read,
"fd_seek": _fd_seek,
"fd_write": _fd_write,
"getTempRet0": _getTempRet0,
"gethostbyaddr": _gethostbyaddr,
"gethostbyname": _gethostbyname,
```

```
"glActiveTexture": _glActiveTexture,
"glAttachShader": _glAttachShader,
"glBeginQuery": _glBeginQuery,
"glBindAttribLocation": _glBindAttribLocation,
"glBindBuffer": _glBindBuffer,
"glBindBufferBase": _glBindBufferBase,
"glBindBufferRange": _glBindBufferRange,
"glBindFramebuffer": _glBindFramebuffer,
"glBindRenderbuffer": _glBindRenderbuffer,
"glBindSampler": _glBindSampler,
"glBindTexture": _glBindTexture,
"glBindVertexArray": _glBindVertexArray,
"glBlendEquation": _glBlendEquation,
"glBlendEquationSeparate": _glBlendEquationSeparate,
"glBlendFuncSeparate": _glBlendFuncSeparate,
"glBlitFramebuffer": _glBlitFramebuffer,
"glBufferData": _glBufferData,
"glBufferSubData": _glBufferSubData,
"glCheckFramebufferStatus": _glCheckFramebufferStatus,
"glClear": _glClear,
"glClearBufferfi": _glClearBufferfi,
"glClearBufferfv": _glClearBufferfv,
"glClearBufferuiv": _glClearBufferuiv,
"glClearColor": _glClearColor,
"glClearDepthf": _glClearDepthf,
"glClearStencil": _glClearStencil,
"glClientWaitSync": _glClientWaitSync,
"glColorMask": _glColorMask,
"glCompileShader": _glCompileShader,
"glCompressedTexImage2D": _glCompressedTexImage2D,
"glCompressedTexImage3D": _glCompressedTexImage3D,
"glCompressedTexSubImage2D": _glCompressedTexSubImage2D,
"glCompressedTexSubImage3D": _glCompressedTexSubImage3D,
"glCopyBufferSubData": _glCopyBufferSubData,
"glCopyTexImage2D": _glCopyTexImage2D,
"glCopyTexSubImage2D": _glCopyTexSubImage2D,
"glCreateProgram": _glCreateProgram,
"glCreateShader": _glCreateShader,
"glCullFace": _glCullFace,
"glDeleteBuffers": _glDeleteBuffers,
"glDeleteFramebuffers": _glDeleteFramebuffers,
"glDeleteProgram": _glDeleteProgram,
"glDeleteQueries": _glDeleteQueries,
"glDeleteRenderbuffers": _glDeleteRenderbuffers,
"glDeleteSamplers": _glDeleteSamplers,
"glDeleteShader": _glDeleteShader,
"glDeleteSync": _glDeleteSync,
"glDeleteTextures": _glDeleteTextures,
"glDeleteVertexArrays": _glDeleteVertexArrays,
"glDepthFunc": _glDepthFunc,
"glDepthMask": _glDepthMask,
"glDetachShader": _glDetachShader,
"glDisable": _glDisable,
"glDisableVertexAttribArray": _glDisableVertexAttribArray,
"glDrawArrays": _glDrawArrays,
```

```
"glDrawArraysInstanced": _glDrawArraysInstanced,
"glDrawBuffers": _glDrawBuffers,
"glDrawElements": _glDrawElements,
"glDrawElementsInstanced": _glDrawElementsInstanced,
"glEnable": _glEnable,
"glEnableVertexAttribArray": _glEnableVertexAttribArray,
"glEndQuery": _glEndQuery,
"glFenceSync": _glFenceSync,
"glFinish": _glFinish,
"glFlush": _glFlush,
"glFlushMappedBufferRange": _glFlushMappedBufferRange,
"glFramebufferRenderbuffer": _glFramebufferRenderbuffer,
"glFramebufferTexture2D": _glFramebufferTexture2D,
"glFramebufferTextureLayer": _glFramebufferTextureLayer,
"glFrontFace": _glFrontFace,
"glGenBuffers": _glGenBuffers,
"glGenFramebuffers": _glGenFramebuffers,
"glGenQueries": _glGenQueries,
"glGenRenderbuffers": _glGenRenderbuffers,
"glGenSamplers": _glGenSamplers,
"glGenTextures": _glGenTextures,
"glGenVertexArrays": _glGenVertexArrays,
"glGenerateMipmap": _glGenerateMipmap,
"glGetActiveAttrib": _glGetActiveAttrib,
"glGetActiveUniform": _glGetActiveUniform,
"glGetActiveUniformBlockName": _glGetActiveUniformBlockName,
"glGetActiveUniformBlockiv": _glGetActiveUniformBlockiv,
"glGetActiveUniformsiv": _glGetActiveUniformsiv,
"glGetAttribLocation": _glGetAttribLocation,
"glGetBufferSubData": _glGetBufferSubData,
"glGetError": _glGetError,
"glGetFramebufferAttachmentParameteriv": _glGetFramebufferAttachmentParameteriv,
"glGetIntegeri_v": _glGetIntegeri_v,
"glGetIntegerv": _glGetIntegerv,
"glGetInternalformativ": _glGetInternalformativ,
"glGetProgramBinary": _glGetProgramBinary,
"glGetProgramInfoLog": _glGetProgramInfoLog,
"glGetProgramiv": _glGetProgramiv,
"glGetQueryObjectuiv": _glGetQueryObjectuiv,
"glGetQueryiv": _glGetQueryiv,
"glGetRenderbufferParameteriv": _glGetRenderbufferParameteriv,
"glGetShaderInfoLog": _glGetShaderInfoLog,
"glGetShaderPrecisionFormat": _glGetShaderPrecisionFormat,
"glGetShaderSource": _glGetShaderSource,
"glGetShaderiv": _glGetShaderiv,
"glGetString": _glGetString,
"glGetStringi": _glGetStringi,
"glGetTexParameteriv": _glGetTexParameteriv,
"glGetUniformBlockIndex": _glGetUniformBlockIndex,
"glGetUniformIndices": _glGetUniformIndices,
"glGetUniformLocation": _glGetUniformLocation,
"glGetUniformiv": _glGetUniformiv,
"glGetVertexAttribiv": _glGetVertexAttribiv,
"glInvalidateFramebuffer": _glInvalidateFramebuffer,
"glIsEnabled": _glIsEnabled,
```

```
"glIsVertexArray": _glIsVertexArray,
"glLinkProgram": _glLinkProgram,
"glMapBufferRange": _glMapBufferRange,
"glPixelStorei": _glPixelStorei,
"glPolygonOffset": _glPolygonOffset,
"glProgramBinary": _glProgramBinary,
"glProgramParameteri": _glProgramParameteri,
"glReadBuffer": _glReadBuffer,
"glReadPixels": _glReadPixels,
"glRenderbufferStorage": _glRenderbufferStorage,
"glRenderbufferStorageMultisample": _glRenderbufferStorageMultisample,
"glSamplerParameteri": _glSamplerParameteri,
"glScissor": _glScissor,
"glShaderSource": _glShaderSource,
"glStencilFuncSeparate": _glStencilFuncSeparate,
"glStencilMask": _glStencilMask,
"glStencilOpSeparate": _glStencilOpSeparate,
"glTexImage2D": _glTexImage2D,
"glTexImage3D": _glTexImage3D,
"glTexParameterf": _glTexParameterf,
"glTexParameterI": _glTexParameterI,
"glTexParameteriv": _glTexParameteriv,
"glTexStorage2D": _glTexStorage2D,
"glTexStorage3D": _glTexStorage3D,
"glTexSubImage2D": _glTexSubImage2D,
"glTexSubImage3D": _glTexSubImage3D,
"glUniform1fv": _glUniform1fv,
"glUniform1i": _glUniform1i,
"glUniform1liv": _glUniform1liv,
"glUniform1uiv": _glUniform1uiv,
"glUniform2fv": _glUniform2fv,
"glUniform2iv": _glUniform2iv,
"glUniform2uiv": _glUniform2uiv,
"glUniform3fv": _glUniform3fv,
"glUniform3iv": _glUniform3iv,
"glUniform3uiv": _glUniform3uiv,
"glUniform4fv": _glUniform4fv,
"glUniform4iv": _glUniform4iv,
"glUniform4uiv": _glUniform4uiv,
"glUniformBlockBinding": _glUniformBlockBinding,
"glUniformMatrix3fv": _glUniformMatrix3fv,
"glUniformMatrix4fv": _glUniformMatrix4fv,
"glUnmapBuffer": _glUnmapBuffer,
"glUseProgram": _glUseProgram,
"glValidateProgram": _glValidateProgram,
"glVertexAttrib4f": _glVertexAttrib4f,
"glVertexAttrib4fv": _glVertexAttrib4fv,
"glVertexAttribIPointer": _glVertexAttribIPointer,
"glVertexAttribPointer": _glVertexAttribPointer,
"glViewport": _glViewport,
"invoke_d": invoke_d,
"invoke_ddd": invoke_ddd,
"invoke_ddddi": invoke_ddddi,
"invoke_dddi": invoke_dddi,
"invoke_ddi": invoke_ddi,
```

"`invoke_ddiii`": `invoke_ddiii`,
"`invoke_dfi`": `invoke_dfi`,
"`invoke_di`": `invoke_di`,
"`invoke_did`": `invoke_did`,
"`invoke_didd`": `invoke_didd`,
"`invoke_didi`": `invoke_didi`,
"`invoke_didii`": `invoke_didii`,
"`invoke_dii`": `invoke_dii`,
"`invoke_didiid`": `invoke_didiid`,
"`invoke_diiii`": `invoke_diiii`,
"`invoke_diiidi`": `invoke_diiidi`,
"`invoke_diiiiid`": `invoke_diiiiid`,
"`invoke_diiiiii`": `invoke_diiiiii`,
"`invoke_dji`": `invoke_dji`,
"`invoke_f`": `invoke_f`,
"`invoke_fdi`": `invoke_fdi`,
"`invoke_ff`": `invoke_ff`,
"`invoke_ffffffffi`": `invoke_ffffffffi`,
"`invoke_fffffi`": `invoke_fffffi`,
"`invoke_ffffi`": `invoke_ffffi`,
"`invoke_fffifi`": `invoke_fffifi`,
"`invoke_ffffififi`": `invoke_ffffififi`,
"`invoke_ffi`": `invoke_ffi`,
"`invoke_ffii`": `invoke_ffii`,
"`invoke_fi`": `invoke_fi`,
"`invoke_fif`": `invoke_fif`,
"`invoke_fiff`": `invoke_fiff`,
"`invoke_fifffi`": `invoke_fifffi`,
"`invoke_fififi`": `invoke_fififi`,
"`invoke_fifi`": `invoke_fifi`,
"`invoke_fifii`": `invoke_fifii`,
"`invoke_fii`": `invoke_fii`,
"`invoke_fiifffi`": `invoke_fiifffi`,
"`invoke_fiifi`": `invoke_fiifi`,
"`invoke_fiififi`": `invoke_fiififi`,
"`invoke_fiiii`": `invoke_fiiii`,
"`invoke_fiiiiii`": `invoke_fiiiiii`,
"`invoke_fji`": `invoke_fji`,
"`invoke_i`": `invoke_i`,
"`invoke_iddi`": `invoke_iddi`,
"`invoke_idi`": `invoke_idi`,
"`invoke_idiiid`": `invoke_idiiid`,
"`invoke_idiiii`": `invoke_idiiii`,
"`invoke_idiiiiid`": `invoke_idiiiiid`,
"`invoke_if`": `invoke_if`,
"`invoke_ifffi`": `invoke_ifffi`,
"`invoke_iffi`": `invoke_iffi`,
"`invoke_ifiii`": `invoke_ifiii`,
"`invoke_ifiiii`": `invoke_ifiiii`,
"`invoke_ii`": `invoke_ii`,

"`invoke_iijiii`": `invoke_iijiii`,
"`invoke_ijiiii`": `invoke_ijiiii`,
"`invoke_ijiiiiii`": `invoke_ijiiiiii`,
"`invoke_ijiji`": `invoke_ijiji`,
"`invoke_ijjjii`": `invoke_ijjjii`,
"`invoke_ij`": `invoke_ij`,
"`invoke_iji`": `invoke_iji`,
"`invoke_ijii`": `invoke_ijii`,
"`invoke_ijiii`": `invoke_ijiii`,
"`invoke_ijiiii`": `invoke_ijiiii`,
"`invoke_ijiiiiji`": `invoke_ijiiiiji`,
"`invoke_ijiiiji`": `invoke_ijiiiji`,
"`invoke_ijiiiji`": `invoke_ijiiiji`,
"`invoke_ijijiiii`": `invoke_ijijiiii`,
"`invoke_ijji`": `invoke_ijji`,
"`invoke_ijjiii`": `invoke_ijjiii`,
"`invoke_ijjjf`": `invoke_ijjjf`,
"`invoke_ijjjfi`": `invoke_ijjjfi`,
"`invoke_ijjiji`": `invoke_ijjiji`,
"`invoke_ijjjiji`": `invoke_ijjjiji`,
"`invoke_j`": `invoke_j`,
"`invoke_jai`": `invoke_jai`,
"`invoke_jdii`": `invoke_jdii`,
"`invoke_jfi`": `invoke_jfi`,
"`invoke_ji`": `invoke_ji`,
"`invoke_jidi`": `invoke_jidi`,
"`invoke_jidii`": `invoke_jidii`,
"`invoke_jii`": `invoke_jii`,
"`invoke_jiii`": `invoke_jiii`,
"`invoke_jiiii`": `invoke_jiiii`,
"`invoke_jiiiiii`": `invoke_jiiiiii`,
"`invoke_jiiiiiiii`": `invoke_jiiiiiiii`,
"`invoke_jiiiiiiiiii`": `invoke_jiiiiiiiiii`,
"`invoke_jiiiiiji`": `invoke_jiiiiiji`,
"`invoke_jiiiji`": `invoke_jiiiji`,
"`invoke_jij`": `invoke_jij`,
"`invoke_jiji`": `invoke_jiji`,
"`invoke_jijiji`": `invoke_jijiji`,
"`invoke_jijiji`": `invoke_jijiji`,
"`invoke_jijjj`": `invoke_jijjj`,
"`invoke_jijiji`": `invoke_jijiji`,
"`invoke_jijjjiji`": `invoke_jijjjiji`,
"`invoke_jiji`": `invoke_jiji`,
"`invoke_jjii`": `invoke_jjii`,
"`invoke_jjiji`": `invoke_jjiji`,
"`invoke_jjiiii`": `invoke_jjiiii`,

"`invoke_viffiiiiiiii`": `invoke_viffiiiiiiii`,
"`invoke_vifiinvoke_vifi`,
"`invoke_vififiinvoke_vififi`,
"`invoke_vifiiinvoke_vifii`,
"`invoke_vifiiiinvoke_vifiii`,
"`invoke_vifiiiiinvoke_vifiiii`,
"`invoke_vifiifinvoke_vifiif`,
"`invoke_viifffiiiiinvoke_viifffiiii`,
"`invoke_viifffiinvoke_viifffi`,
"`invoke_viiiffiinvoke_viiiffi`,
"`invoke_viiiffiiinvoke_viiiffii`,
"`invoke_viiiffiiiiinvoke_viiiffiiii`,
"`invoke_viiiffiiiiiiinvoke_viiiffiiiiii`,
"`invoke_viiifiinvoke_viiifi`,
"`invoke_viiififiinvoke_viiififi`,
"`invoke_viiifiiinvoke_viiifii`,
"`invoke_viiifiiiinvoke_viiifiii`,
"`invoke_viiiiinvoke_viiii`,
"`invoke_viiidiinvoke_viiidi`,
"`invoke_viididiiinvoke_viididii`,
"`invoke_viiiffiinvoke_viiiffi`,
"`invoke_viiiffiiinvoke_viiiffii`,
"`invoke_viiiiififiinvoke_viiiiififi`,
"`invoke_viiiiiiinvoke_viiiiii`,
"`invoke_viiiiiffiinvoke_viiiiiffi`,
"`invoke_viiiiifiinvoke_viiiiifi`,
"`invoke_viiiiiiinvoke_viiiiii`,
"`invoke_viiiiififiinvoke_viiiiififi`,
"`invoke_viiiiiiiiinvoke_viiiiiiii`,
"`invoke_viiiiiiiiiiinvoke_viiiiiiiiii`,
"`invoke_viiiiiiiiiiiiinvoke_viiiiiiiiiiii`,
"`invoke_viiiiiiiiiiiiiiinvoke_viiiiiiiiiiiiii`,
"`invoke_viiiiiiiiiiiiiiiiinvoke_viiiiiiiiiiiiiiii`,
"`invoke_viiiiiiijiiinvoke_viiiiiiijii`,
"`invoke_viiiiijijiinvoke_viiiiijiji`,
"`invoke_viiijijiinvoke_viiijiji`,
"`invoke_viiijiinvoke_viiiji`,
"`invoke_viiijijiinvoke_viiijiji`,
"`invoke_viiijijiinvoke_viiijiji`,
"`invoke_vijjinvoke_vijj`,
"`invoke_vijiinvoke_viji`,
"`invoke_vijijiinvoke_vijiji`,
"`invoke_vijiiiiinvoke_vijiiii`,
"`invoke_vijiiiiiinvoke_vijiiiii`,
"`invoke_vijijijiinvoke_vijijiji`,
"`invoke_vijijijiinvoke_vijijiji`,
"`invoke_vijijijiinvoke_vijijiji`,
"`invoke_vijijijiinvoke_vijijiji`

```

"invoke_vijjji": invoke_vijjji,
"invoke_vij": invoke_vij,
"invoke_vijfff": invoke_vijfff,
"invoke_vijffffi": invoke_vijffffi,
"invoke_viji": invoke_viji,
"invoke_vijiii": invoke_vijiii,
"invoke_vijiiii": invoke_vijiiii,
"invoke_vijiiiiii": invoke_vijiiiiii,
"invoke_vijiiiiiii": invoke_vijiiiiiii,
"invoke_vijiiiiiiii": invoke_vijiiiiiiii,
"invoke_vijiji": invoke_vijiji,
"invoke_vijijji": invoke_vijijji,
"invoke_vijiji": invoke_vijiji,
"invoke_vijiji": invoke_vijiji,
"invoke_vj": invoke_vj,
"invoke_vji": invoke_vji,
"invoke_vjii": invoke_vjii,
"invoke_vjiii": invoke_vjiii,
"invoke_vjiiii": invoke_vjiiii,
"invoke_vjiiiiii": invoke_vjiiiiii,
"invoke_vjjjii": invoke_vjjjii,
"invoke_vjjjiiii": invoke_vjjjiiii,
"invoke_vjjjiiiii": invoke_vjjjiiiii,
"llvm_eh_typeid_for": _llvm_eh_typeid_for,
"setTempRet0": _setTempRet0,
"strftime": _strftime
};

var asm = createWasm();
/** @type {function(...*)?:?} */
var __wasm_call_ctors = Module["__wasm_call_ctors"] =
createExportWrapper("__wasm_call_ctors");

/** @type {function(...*)?:?} */
var _getMemInfo = Module["_getMemInfo"] = createExportWrapper("getMemInfo");

/** @type {function(...*)?:?} */
var _SendMessageFloat = Module["_SendMessageFloat"] =
createExportWrapper("SendMessageFloat");

/** @type {function(...*)?:?} */
var _SendMessageString = Module["_SendMessageString"] =
createExportWrapper("SendMessageString");

/** @type {function(...*)?:?} */
var _SendMessage = Module["_SendMessage"] = createExportWrapper("SendMessage");

/** @type {function(...*)?:?} */
var _SetFullscreen = Module["_SetFullscreen"] = createExportWrapper("SetFullscreen");

/** @type {function(...*)?:?} */
var _main = Module["_main"] = createExportWrapper("main");

/** @type {function(...*)?:?} */

```

```

var _InjectProfilerSample = Module["_InjectProfilerSample"] =
createExportWrapper("InjectProfilerSample");

/** @type {function(...*)?:?} */
var __errno_location = Module["__errno_location"] =
createExportWrapper("__errno_location");

/** @type {function(...*)?:?} */
var __stdio_exit = Module["__stdio_exit"] = createExportWrapper("__stdio_exit");

/** @type {function(...*)?:?} */
var __dl_seterr = Module["__dl_seterr"] = createExportWrapper("__dl_seterr");

/** @type {function(...*)?:?} */
var _htonl = Module["_htonl"] = createExportWrapper("htonl");

/** @type {function(...*)?:?} */
var _htons = Module["_htons"] = createExportWrapper("htons");

/** @type {function(...*)?:?} */
var _ntohs = Module["_ntohs"] = createExportWrapper("ntohs");

/** @type {function(...*)?:?} */
var _strlen = Module["_strlen"] = createExportWrapper("strlen");

/** @type {function(...*)?:?} */
var _malloc = Module["_malloc"] = createExportWrapper("malloc");

/** @type {function(...*)?:?} */
var _free = Module["_free"] = createExportWrapper("free");

/** @type {function(...*)?:?} */
var _emscripten_builtin_memalign = Module["_emscripten_builtin_memalign"] =
createExportWrapper("emscripten_builtin_memalign");

/** @type {function(...*)?:?} */
var _setThrew = Module["_setThrew"] = createExportWrapper("setThrew");

/** @type {function(...*)?:?} */
var _saveSetjmp = Module["_saveSetjmp"] = createExportWrapper("saveSetjmp");

/** @type {function(...*)?:?} */
var _emscripten_stack_init = Module["_emscripten_stack_init"] = function() {
  return (_emscripten_stack_init = Module["_emscripten_stack_init"] =
Module["asm"]["emscripten_stack_init"]).apply(null, arguments);
};

/** @type {function(...*)?:?} */
var _emscripten_stack_get_free = Module["_emscripten_stack_get_free"] = function() {
  return (_emscripten_stack_get_free = Module["_emscripten_stack_get_free"] =
Module["asm"]["emscripten_stack_get_free"]).apply(null, arguments);
};

/** @type {function(...*)?:?} */
var _emscripten_stack_get_base = Module["_emscripten_stack_get_base"] = function() {

```

```

    return (_emscripten_stack_get_base = Module["_emscripten_stack_get_base"] =
Module["asm"]["emscripten_stack_get_base"]).apply(null, arguments);
};

/** @type {function(...*):?} */
var _emscripten_stack_get_end = Module["_emscripten_stack_get_end"] = function() {
    return (_emscripten_stack_get_end = Module["_emscripten_stack_get_end"] =
Module["asm"]["emscripten_stack_get_end"]).apply(null, arguments);
};

/** @type {function(...*):?} */
var stackSave = Module["stackSave"] = createExportWrapper("stackSave");

/** @type {function(...*):?} */
var stackRestore = Module["stackRestore"] = createExportWrapper("stackRestore");

/** @type {function(...*):?} */
var stackAlloc = Module["stackAlloc"] = createExportWrapper("stackAlloc");

/** @type {function(...*):?} */
var __cxa_demangle = Module["__cxa_demangle"] =
createExportWrapper("__cxa_demangle");

/** @type {function(...*):?} */
var __cxa_can_catch = Module["__cxa_can_catch"] =
createExportWrapper("__cxa_can_catch");

/** @type {function(...*):?} */
var __cxa_is_pointer_type = Module["__cxa_is_pointer_type"] =
createExportWrapper("__cxa_is_pointer_type");

/** @type {function(...*):?} */
var dynCall_iidiiii = Module["dynCall_iidiiii"] =
createExportWrapper("dynCall_iidiiii");

/** @type {function(...*):?} */
var dynCall_vii = Module["dynCall_vii"] = createExportWrapper("dynCall_vii");

/** @type {function(...*):?} */
var dynCall_iiii = Module["dynCall_iiii"] = createExportWrapper("dynCall_iiii");

/** @type {function(...*):?} */
var dynCall_v = Module["dynCall_v"] = createExportWrapper("dynCall_v");

/** @type {function(...*):?} */
var dynCall_vi = Module["dynCall_vi"] = createExportWrapper("dynCall_vi");

/** @type {function(...*):?} */
var dynCall_viii = Module["dynCall_viii"] = createExportWrapper("dynCall_viii");

/** @type {function(...*):?} */
var dynCall_jiji = Module["dynCall_jiji"] = createExportWrapper("dynCall_jiji");

/** @type {function(...*):?} */
var dynCall_ii = Module["dynCall_ii"] = createExportWrapper("dynCall_ii");

```

```
/** @type {function(...*:?) *} */
var dynCall_iieee = Module["dynCall_ieeee"] = createExportWrapper("dynCall_ieeee");

/** @type {function(...*:?) *} */
var dynCall_iiie = Module["dynCall_iiie"] = createExportWrapper("dynCall_iiie");

/** @type {function(...*:?) *} */
var dynCall_viiii = Module["dynCall_viiii"] = createExportWrapper("dynCall_viiii");

/** @type {function(...*:?) *} */
var dynCall_viiiiii = Module["dynCall_viiiiii"] =
createExportWrapper("dynCall_viiiiii");

/** @type {function(...*:?) *} */
var dynCall_viiiiii = Module["dynCall_viiiiii"] = createExportWrapper("dynCall_viiiiii");

/** @type {function(...*:?) *} */
var dynCall_i = Module["dynCall_i"] = createExportWrapper("dynCall_i");

/** @type {function(...*:?) *} */
var dynCall_iiiiiii = Module["dynCall_iiiiiii"] =
createExportWrapper("dynCall_iiiiiii");

/** @type {function(...*:?) *} */
var dynCall_iijjiji = Module["dynCall_ijjjiji"] =
createExportWrapper("dynCall_ijjjiji");

/** @type {function(...*:?) *} */
var dynCall_iij = Module["dynCall_ijj"] = createExportWrapper("dynCall_ijj");

/** @type {function(...*:?) *} */
var dynCall_iiiiiii = Module["dynCall_iiiiiii"] =
createExportWrapper("dynCall_iiiiiii");

/** @type {function(...*:?) *} */
var dynCall_jii = Module["dynCall_jii"] = createExportWrapper("dynCall_jii");

/** @type {function(...*:?) *} */
var dynCall_viiiiii = Module["dynCall_viiiiii"] =
createExportWrapper("dynCall_viiiiii");

/** @type {function(...*:?) *} */
var dynCall_vijii = Module["dynCall_vijii"] = createExportWrapper("dynCall_vijii");

/** @type {function(...*:?) *} */
var dynCall_iififi = Module["dynCall_iififi"] = createExportWrapper("dynCall_iififi");

/** @type {function(...*:?) *} */
var dynCall_vififi = Module["dynCall_vififi"] = createExportWrapper("dynCall_vififi");

/** @type {function(...*:?) *} */
```

```
var dynCall_iiijiji = Module["dynCall_iiijiji"] =
createExportWrapper("dynCall_iiijiji");

/** @type {function(...*)?:} */
var dynCall_vidi = Module["dynCall_vidi"] = createExportWrapper("dynCall_vidi");

/** @type {function(...*)?:} */
var dynCall_viidi = Module["dynCall_viidi"] = createExportWrapper("dynCall_viidi");

/** @type {function(...*)?:} */
var dynCall_iiidii = Module["dynCall_iiidii"] =
createExportWrapper("dynCall_iiidii");

/** @type {function(...*)?:} */
var dynCall_iidiii = Module["dynCall_iidiii"] = createExportWrapper("dynCall_iidiii");

/** @type {function(...*)?:} */
var dynCall_iidi = Module["dynCall_iidi"] = createExportWrapper("dynCall_iidi");

/** @type {function(...*)?:} */
var dynCall_viji = Module["dynCall_viji"] = createExportWrapper("dynCall_viji");

/** @type {function(...*)?:} */
var dynCall_viiji = Module["dynCall_viiji"] = createExportWrapper("dynCall_viiji");

/** @type {function(...*)?:} */
var dynCall_ijjiii = Module["dynCall_ijjiii"] = createExportWrapper("dynCall_ijjiii");

/** @type {function(...*)?:} */
var dynCall_iiji = Module["dynCall_ijji"] = createExportWrapper("dynCall_ijji");

/** @type {function(...*)?:} */
var dynCall_iiiiiiiiiii = Module["dynCall_iiiiiiiiiii"] =
createExportWrapper("dynCall_iiiiiiiiiii");

/** @type {function(...*)?:} */
var dynCall_iiiiiiiiiii = Module["dynCall_iiiiiiiiiii"] =
createExportWrapper("dynCall_iiiiiiiiiii");

/** @type {function(...*)?:} */
var dynCall_fii = Module["dynCall_fii"] = createExportWrapper("dynCall_fii");

/** @type {function(...*)?:} */
var dynCall_vifi = Module["dynCall_vifi"] = createExportWrapper("dynCall_vifi");

/** @type {function(...*)?:} */
var dynCall_fiii = Module["dynCall_fiii"] = createExportWrapper("dynCall_fiii");

/** @type {function(...*)?:} */
var dynCall_iiiiiiiii = Module["dynCall_iiiiiiiii"] =
createExportWrapper("dynCall_iiiiiiiii");

/** @type {function(...*)?:} */
var dynCall_iiiiiiiii = Module["dynCall_iiiiiiiii"] =
createExportWrapper("dynCall_iiiiiiiii");
```

```
/** @type {function(...*:?) *} */
var dynCall_iiiiiiii = Module["dynCall_iiiiiiii"] =
createExportWrapper("dynCall_iiiiiiii");

/** @type {function(...*:?) *} */
var dynCall_vifffffi = Module["dynCall_vifffffi"] =
createExportWrapper("dynCall_vifffffi");

/** @type {function(...*:?) *} */
var dynCall_viiififi = Module["dynCall_viiififi"] =
createExportWrapper("dynCall_viiififi");

/** @type {function(...*:?) *} */
var dynCall_ijiji = Module["dynCall_ijiji"] = createExportWrapper("dynCall_ijiji");

/** @type {function(...*:?) *} */
var dynCall_iifi = Module["dynCall_iifi"] = createExportWrapper("dynCall_iifi");

/** @type {function(...*:?) *} */
var dynCall_ji = Module["dynCall_ji"] = createExportWrapper("dynCall_ji");

/** @type {function(...*:?) *} */
var dynCall_dii = Module["dynCall_dii"] = createExportWrapper("dynCall_dii");

/** @type {function(...*:?) *} */
var dynCall_diii = Module["dynCall_diii"] = createExportWrapper("dynCall_diii");

/** @type {function(...*:?) *} */
var dynCall_jiii = Module["dynCall_jiii"] = createExportWrapper("dynCall_jiii");

/** @type {function(...*:?) *} */
var dynCall_viiiiiii = Module["dynCall_viiiiiii"] =
createExportWrapper("dynCall_viiiiiii");

/** @type {function(...*:?) *} */
var dynCall_viiiiifi = Module["dynCall_viiiiifi"] =
createExportWrapper("dynCall_viiiiifi");

/** @type {function(...*:?) *} */
var dynCall_viiffffffiiii = Module["dynCall_viiffffffiiii"] =
createExportWrapper("dynCall_viiffffffiiii");

/** @type {function(...*:?) *} */
var dynCall_viffiiii = Module["dynCall_viffiiii"] =
createExportWrapper("dynCall_viffiiii");

/** @type {function(...*:?) *} */
var dynCall_viififi = Module["dynCall_viififi"] = createExportWrapper("dynCall_viififi");

/** @type {function(...*:?) *} */
var dynCall_viiifffiiii = Module["dynCall_viiifffiiii"] =
createExportWrapper("dynCall_viiifffiiii");

/** @type {function(...*:?) *} */
var dynCall_ifffi = Module["dynCall_ifffi"] = createExportWrapper("dynCall_ifffi");
```

```
/** @type {function(...*:?) *} */
var dynCall_fiiii = Module["dynCall_fiiii"] = createExportWrapper("dynCall_fiiii");

/** @type {function(...*:?) *} */
var dynCall_viffffi = Module["dynCall_viffffi"] =
createExportWrapper("dynCall_viffffi");

/** @type {function(...*:?) *} */
var dynCall_viffffi = Module["dynCall_viffffi"] = createExportWrapper("dynCall_viffffi");

/** @type {function(...*:?) *} */
var dynCall_fiiiii = Module["dynCall_fiiiii"] = createExportWrapper("dynCall_fiiiii");

/** @type {function(...*:?) *} */
var dynCall_viiiiiiiiiiii = Module["dynCall_viiiiiiiiiiii"] =
createExportWrapper("dynCall_viiiiiiiiiiii");

/** @type {function(...*:?) *} */
var dynCall_vifiiiiii = Module["dynCall_vifiiiiii"] =
createExportWrapper("dynCall_vifiiiiii");

/** @type {function(...*:?) *} */
var dynCall_viiiiiiii = Module["dynCall_viiiiiiii"] =
createExportWrapper("dynCall_viiiiiiii");

/** @type {function(...*:?) *} */
var dynCall_viiiffiiiiiiii = Module["dynCall_viiiffiiiiiiii"] =
createExportWrapper("dynCall_viiiffiiiiiiii");

/** @type {function(...*:?) *} */
var dynCall_viiiffiiiiii = Module["dynCall_viiiffiiiiii"] =
createExportWrapper("dynCall_viiiffiiiiii");

/** @type {function(...*:?) *} */
var dynCall_viiifi = Module["dynCall_viiifi"] = createExportWrapper("dynCall_viiifi");

/** @type {function(...*:?) *} */
var dynCall_fffffi = Module["dynCall_fffffi"] = createExportWrapper("dynCall_fffffi");

/** @type {function(...*:?) *} */
var dynCall_jdi = Module["dynCall_jdi"] = createExportWrapper("dynCall_jdi");

/** @type {function(...*:?) *} */
var dynCall_viiiji = Module["dynCall_viiiji"] = createExportWrapper("dynCall_viiiji");

/** @type {function(...*:?) *} */
var dynCall_jji = Module["dynCall_jji"] = createExportWrapper("dynCall_jji");

/** @type {function(...*:?) *} */
var dynCall_vji = Module["dynCall_vji"] = createExportWrapper("dynCall_vji");
```

```
/** @type {function(...*)?:?} */
var dynCall_ijji = Module["dynCall_ijji"] = createExportWrapper("dynCall_ijji");

/** @type {function(...*)?:?} */
var dynCall_viiiiiiiiii = Module["dynCall_viiiiiiiiii"] =
createExportWrapper("dynCall_viiiiiiiiii");

/** @type {function(...*)?:?} */
var dynCall_j = Module["dynCall_j"] = createExportWrapper("dynCall_j");

/** @type {function(...*)?:?} */
var dynCall_iiiji = Module["dynCall_iiiji"] = createExportWrapper("dynCall_iiiji");

/** @type {function(...*)?:?} */
var dynCall_viiiiiiiiiiii = Module["dynCall_viiiiiiiiiiii"] =
createExportWrapper("dynCall_viiiiiiiiiiii");

/** @type {function(...*)?:?} */
var dynCall_vijiji = Module["dynCall_vijiji"] = createExportWrapper("dynCall_vijiji");

/** @type {function(...*)?:?} */
var dynCall_ijiii = Module["dynCall_ijiii"] = createExportWrapper("dynCall_ijiii");

/** @type {function(...*)?:?} */
var dynCall_ijiiii = Module["dynCall_ijiiii"] = createExportWrapper("dynCall_ijiiii");

/** @type {function(...*)?:?} */
var dynCall_dddi = Module["dynCall_dddi"] = createExportWrapper("dynCall_dddi");

/** @type {function(...*)?:?} */
var dynCall_jiiiii = Module["dynCall_jiiiii"] = createExportWrapper("dynCall_jiiiii");

/** @type {function(...*)?:?} */
var dynCall_iiiiiiiiji = Module["dynCall_iiiiiiiiji"] =
createExportWrapper("dynCall_iiiiiiiiji");

/** @type {function(...*)?:?} */
var dynCall_viiijii = Module["dynCall_viiijii"] =
createExportWrapper("dynCall_viiijii");

/** @type {function(...*)?:?} */
var dynCall_ifi = Module["dynCall_ifi"] = createExportWrapper("dynCall_ifi");

/** @type {function(...*)?:?} */
var dynCall_fi = Module["dynCall_fi"] = createExportWrapper("dynCall_fi");

/** @type {function(...*)?:?} */
var dynCall_viffffii = Module["dynCall_viffffii"] =
createExportWrapper("dynCall_viffffii");

/** @type {function(...*)?:?} */
var dynCall_viffi = Module["dynCall_viffi"] = createExportWrapper("dynCall_viffi");

/** @type {function(...*)?:?} */
```

```
var dynCall_viiififii = Module["dynCall_viiififii"] =
createExportWrapper("dynCall_viiififii");

/** @type {function(...*):?} */
var dynCall_ddi = Module["dynCall_ddi"] = createExportWrapper("dynCall_ddi");

/** @type {function(...*):?} */
var dynCall_fifi = Module["dynCall_fifi"] = createExportWrapper("dynCall_fifi");

/** @type {function(...*):?} */
var dynCall_fdi = Module["dynCall_fdi"] = createExportWrapper("dynCall_fdi");

/** @type {function(...*):?} */
var dynCall_vijji = Module["dynCall_vijji"] = createExportWrapper("dynCall_vijji");

/** @type {function(...*):?} */
var dynCall_iiiifi = Module["dynCall_iiiifi"] = createExportWrapper("dynCall_iiiifi");

/** @type {function(...*):?} */
var dynCall_fifffi = Module["dynCall_fifffi"] = createExportWrapper("dynCall_fifffi");

/** @type {function(...*):?} */
var dynCall_viiffifi = Module["dynCall_viiffifi"] =
createExportWrapper("dynCall_viiffifi");

/** @type {function(...*):?} */
var dynCall_ddddi = Module["dynCall_ddddi"] = createExportWrapper("dynCall_ddddi");

/** @type {function(...*):?} */
var dynCall_ddiii = Module["dynCall_ddiii"] = createExportWrapper("dynCall_ddiii");

/** @type {function(...*):?} */
var dynCall_fffi = Module["dynCall_fffi"] = createExportWrapper("dynCall_fffi");

/** @type {function(...*):?} */
var dynCall_fffi = Module["dynCall_fffi"] = createExportWrapper("dynCall_fffi");

/** @type {function(...*):?} */
var dynCall_jjjji = Module["dynCall_jjjji"] = createExportWrapper("dynCall_jjjji");

/** @type {function(...*):?} */
var dynCall_ijji = Module["dynCall_ijji"] = createExportWrapper("dynCall_ijji");

/** @type {function(...*):?} */
var dynCall_jijji = Module["dynCall_jijji"] = createExportWrapper("dynCall_jijji");

/** @type {function(...*):?} */
var dynCall_if ffi = Module["dynCall_if ffi"] = createExportWrapper("dynCall_if ffi");

/** @type {function(...*):?} */
var dynCall_ffii = Module["dynCall_ffii"] = createExportWrapper("dynCall_ffii");

/** @type {function(...*):?} */
var dynCall_vifii = Module["dynCall_vifii"] = createExportWrapper("dynCall_vifii");
```

```
/** @type {function(...*)?:?} */
var dynCall_iifi = Module["dynCall_iifi"] = createExportWrapper("dynCall_iifi");

/** @type {function(...*)?:?} */
var dynCall_iifii = Module["dynCall_iifii"] = createExportWrapper("dynCall_iifii");

/** @type {function(...*)?:?} */
var dynCall_iisiiffiii = Module["dynCall_iisiiffiii"] =
createExportWrapper("dynCall_iisiiffiii");

/** @type {function(...*)?:?} */
var dynCall_viiifffiii = Module["dynCall_viiifffiii"] =
createExportWrapper("dynCall_viiifffiii");

/** @type {function(...*)?:?} */
var dynCall_viifffffi = Module["dynCall_viifffffi"] =
createExportWrapper("dynCall_viifffffi");

/** @type {function(...*)?:?} */
var dynCall_viffiiii = Module["dynCall_viffiiii"] =
createExportWrapper("dynCall_viffiiii");

/** @type {function(...*)?:?} */
var dynCall_viffififi = Module["dynCall_viffififi"] =
createExportWrapper("dynCall_viffififi");

/** @type {function(...*)?:?} */
var dynCall_ffffifffi = Module["dynCall_ffffifffi"] =
createExportWrapper("dynCall_ffffifffi");

/** @type {function(...*)?:?} */
var dynCall_ffffffffi = Module["dynCall_ffffffffi"] =
createExportWrapper("dynCall_ffffffffi");

/** @type {function(...*)?:?} */
var dynCall_iddi = Module["dynCall_iddi"] = createExportWrapper("dynCall_iddi");

/** @type {function(...*)?:?} */
var dynCall_viiffi = Module["dynCall_viiffi"] = createExportWrapper("dynCall_viiffi");

/** @type {function(...*)?:?} */
var dynCall_viiifiii = Module["dynCall_viiifiii"] =
createExportWrapper("dynCall_viiifiii");

/** @type {function(...*)?:?} */
var dynCall_iiiifiii = Module["dynCall_iiiifiii"] =
createExportWrapper("dynCall_iiiifiii");

/** @type {function(...*)?:?} */
var dynCall_viiifii = Module["dynCall_viiifii"] =
createExportWrapper("dynCall_viiifii");

/** @type {function(...*)?:?} */
var dynCall_iiiifii = Module["dynCall_iiiifii"] =
createExportWrapper("dynCall_iiiifii");
```

```

/** @type {function(...*:?)} */
var dynCall_iifiii = Module["dynCall_iifiii"] =
createExportWrapper("dynCall_iifiii");

/** @type {function(...*:?)} */
var dynCall_iifiifii = Module["dynCall_iifiifii"] =
createExportWrapper("dynCall_iifiifii");

/** @type {function(...*:?)} */
var dynCall_iifiiii = Module["dynCall_iifiiii"] =
createExportWrapper("dynCall_iifiiii");

/** @type {function(...*:?)} */
var dynCall_iiiiiiffiiiiiiiiiiiiii = Module["dynCall_iiiiiiffiiiiiiiiiiiiii"] =
createExportWrapper("dynCall_iiiiiiffiiiiiiiiiiiiii");

/** @type {function(...*:?)} */
var dynCall_iiiiiiffiiiiiiiiifffiiii = Module["dynCall_iiiiiiffiiiiiiiiifffiiii"] =
createExportWrapper("dynCall_iiiiiiffiiiiiiiiifffiiii");

/** @type {function(...*:?)} */
var dynCall_fifii = Module["dynCall_fifii"] = createExportWrapper("dynCall_fifii");

/** @type {function(...*:?)} */
var dynCall_fiifi = Module["dynCall_fiifi"] = createExportWrapper("dynCall_fiifi");

/** @type {function(...*:?)} */
var dynCall_viiiffi = Module["dynCall_viiiffi"] =
createExportWrapper("dynCall_viiiffi");

/** @type {function(...*:?)} */
var dynCall_viiiffi = Module["dynCall_viiiffi"] =
createExportWrapper("dynCall_viiiffi");

/** @type {function(...*:?)} */
var dynCall_vifffii = Module["dynCall_vifffii"] =
createExportWrapper("dynCall_vifffii");

/** @type {function(...*:?)} */
var dynCall_fiifii = Module["dynCall_fiifii"] = createExportWrapper("dynCall_fiifii");

/** @type {function(...*:?)} */
var dynCall_viiiiiffi = Module["dynCall_viiiiiffi"] =
createExportWrapper("dynCall_viiiiiffi");

/** @type {function(...*:?)} */
var dynCall_viifiii = Module["dynCall_viifiii"] =
createExportWrapper("dynCall_viifiii");

/** @type {function(...*:?)} */
var dynCall_vijiji = Module["dynCall_vijiji"] = createExportWrapper("dynCall_vijiji");

/** @type {function(...*:?)} */

```

```
var dynCall_vijiiji = Module["dynCall_vijiiji"] =
createExportWrapper("dynCall_vijiiji");

/** @type {function(...*)?:?} */
var dynCall_vijiiji = Module["dynCall_vijiiji"];
createExportWrapper("dynCall_vijiiji");

/** @type {function(...*)?:?} */
var dynCall_jiiiiii = Module["dynCall_jiiiiii"] =
createExportWrapper("dynCall_jiiiiii");

/** @type {function(...*)?:?} */
var dynCall_jiji = Module["dynCall_jiji"] = createExportWrapper("dynCall_jiji");

/** @type {function(...*)?:?} */
var dynCall_ijii = Module["dynCall_ijii"] = createExportWrapper("dynCall_ijii");

/** @type {function(...*)?:?} */
var dynCall_viiiiiiiiiiii = Module["dynCall_viiiiiiiiiiii"] =
createExportWrapper("dynCall_viiiiiiiiiiii");

/** @type {function(...*)?:?} */
var dynCall_viiifi = Module["dynCall_viiifi"] =
createExportWrapper("dynCall_viiifi");

/** @type {function(...*)?:?} */
var dynCall_iiffiiiiii = Module["dynCall_iiffiiiiii"] =
createExportWrapper("dynCall_iiffiiiiii");

/** @type {function(...*)?:?} */
var dynCall_fiiiffi = Module["dynCall_fiiiffi"] =
createExportWrapper("dynCall_fiiiffi");

/** @type {function(...*)?:?} */
var dynCall_fiiffi = Module["dynCall_fiiffi"] = createExportWrapper("dynCall_fiiffi");

/** @type {function(...*)?:?} */
var dynCall_vffi = Module["dynCall_vffi"] = createExportWrapper("dynCall_vffi");

/** @type {function(...*)?:?} */
var dynCall_fiiiifi = Module["dynCall_fiiiifi"] =
createExportWrapper("dynCall_fiiiifi");

/** @type {function(...*)?:?} */
var dynCall_diiiidi = Module["dynCall_diiiidi"] =
createExportWrapper("dynCall_diiiidi");

/** @type {function(...*)?:?} */
var dynCall_didi = Module["dynCall_didi"] = createExportWrapper("dynCall_didi");

/** @type {function(...*)?:?} */
var dynCall_jiiiji = Module["dynCall_jiiiji"] =
createExportWrapper("dynCall_jiiiji");

/** @type {function(...*)?:?} */
```

```
var dynCall_viffiiiiiiii = Module["dynCall_viffiiiiiiii"] =
createExportWrapper("dynCall_viffiiiiiiii");

/** @type {function(...*):?} */
var dynCall_viffiiiiii = Module["dynCall_viffiiiiii"];
createExportWrapper("dynCall_viffiiiiii");

/** @type {function(...*):?} */
var dynCall_viidiji = Module["dynCall_viidiji"];
createExportWrapper("dynCall_viidiji");

/** @type {function(...*):?} */
var dynCall_viidjii = Module["dynCall_viidjii"];
createExportWrapper("dynCall_viidjii");

/** @type {function(...*):?} */
var dynCall_vfiii = Module["dynCall_vfiii"] = createExportWrapper("dynCall_vfiii");

/** @type {function(...*):?} */
var dynCall_vfii = Module["dynCall_vfii"] = createExportWrapper("dynCall_vfii");

/** @type {function(...*):?} */
var dynCall_idi = Module["dynCall_idi"] = createExportWrapper("dynCall_idi");

/** @type {function(...*):?} */
var dynCall_iiifiiiiii = Module["dynCall_iiifiiiiii"] =
createExportWrapper("dynCall_iiifiiiiii");

/** @type {function(...*):?} */
var dynCall_iiifiiii = Module["dynCall_iiifiiii"] =
createExportWrapper("dynCall_iiifiiii");

/** @type {function(...*):?} */
var dynCall_viiidi = Module["dynCall_viiidi"] = createExportWrapper("dynCall_viiidi");

/** @type {function(...*):?} */
var dynCall_di = Module["dynCall_di"] = createExportWrapper("dynCall_di");

/** @type {function(...*):?} */
var dynCall_viiiidi = Module["dynCall_viiiidi"] =
createExportWrapper("dynCall_viiiidi");

/** @type {function(...*):?} */
var dynCall_jiiii = Module["dynCall_jiiii"] = createExportWrapper("dynCall_jiiii");

/** @type {function(...*):?} */
var dynCall_iiiiiidii = Module["dynCall_iiiiiidii"] =
createExportWrapper("dynCall_iiiiiidii");

/** @type {function(...*):?} */
var dynCall_viidii = Module["dynCall_viidii"] = createExportWrapper("dynCall_viidii");

/** @type {function(...*):?} */
var dynCall_vijiji = Module["dynCall_vijiji"] = createExportWrapper("dynCall_vijiji");

/** @type {function(...*):?} */
```

```

var dynCall_iiffi = Module["dynCall_iiffi"] = createExportWrapper("dynCall_iiffi");
/** @type {function(...*:?) *} */
var dynCall_fiiiiiii = Module["dynCall_fiiiiiii"] =
createExportWrapper("dynCall_fiiiiiii");

/** @type {function(...*:?) *} */
var dynCall_vijiji = Module["dynCall_vijiji"] =
createExportWrapper("dynCall_vijiji");

/** @type {function(...*:?) *} */
var dynCall_iijji = Module["dynCall_iijji"] = createExportWrapper("dynCall_iijji");
/** @type {function(...*:?) *} */
var dynCall_vfi = Module["dynCall_vfi"] = createExportWrapper("dynCall_vfi");

/** @type {function(...*:?) *} */
var dynCall_viidjii = Module["dynCall_viidjii"] =
createExportWrapper("dynCall_viidjii");

/** @type {function(...*:?) *} */
var dynCall_fffffi = Module["dynCall_fffffi"] = createExportWrapper("dynCall_fffffi");

/** @type {function(...*:?) *} */
var dynCall_vjii = Module["dynCall_vjii"] = createExportWrapper("dynCall_vjii");

/** @type {function(...*:?) *} */
var dynCall_viiiiiijiiii = Module["dynCall_viiiiiijiiii"] =
createExportWrapper("dynCall_viiiiiijiiii");

/** @type {function(...*:?) *} */
var dynCall_vfffffi = Module["dynCall_vfffffi"] = createExportWrapper("dynCall_vfffffi");

/** @type {function(...*:?) *} */
var dynCall_vifffffi = Module["dynCall_vifffffi"] =
createExportWrapper("dynCall_vifffffi");

/** @type {function(...*:?) *} */
var dynCall_vifffffffi = Module["dynCall_vifffffffi"] =
createExportWrapper("dynCall_vifffffffi");

/** @type {function(...*:?) *} */
var dynCall_viiiiifiiiiii = Module["dynCall_viiiiifiiiiii"] =
createExportWrapper("dynCall_viiiiifiiiiii");

/** @type {function(...*:?) *} */
var dynCall_vfffffi = Module["dynCall_vfffffi"] = createExportWrapper("dynCall_vfffffi");

/** @type {function(...*:?) *} */
var dynCall_viififii = Module["dynCall_viififii"] =
createExportWrapper("dynCall_viififii");

/** @type {function(...*:?) *} */
var dynCall_fiffi = Module["dynCall_fiffi"] = createExportWrapper("dynCall_fiffi");

```

```
/** @type {function(...*:?) *} */
var dynCall_viffii = Module["dynCall_viffii"] = createExportWrapper("dynCall_viffii");

/** @type {function(...*:?) *} */
var dynCall_diidi = Module["dynCall_diidi"] = createExportWrapper("dynCall_diidi");

/** @type {function(...*:?) *} */
var dynCall_jiiji = Module["dynCall_jiiji"] = createExportWrapper("dynCall_jiiji");

/** @type {function(...*:?) *} */
var dynCall_iidiii = Module["dynCall_iidiii"] = createExportWrapper("dynCall_iidiii");

/** @type {function(...*:?) *} */
var dynCall_viiidii = Module["dynCall_viiidii"] =
createExportWrapper("dynCall_viiidii");

/** @type {function(...*:?) *} */
var dynCall_vidiiii = Module["dynCall_vidiiii"] =
createExportWrapper("dynCall_vidiiii");

/** @type {function(...*:?) *} */
var dynCall_diiii = Module["dynCall_diiii"] = createExportWrapper("dynCall_diiii");

/** @type {function(...*:?) *} */
var dynCall_iiijii = Module["dynCall_iiijii"] = createExportWrapper("dynCall_iiijii");

/** @type {function(...*:?) *} */
var dynCall_iiddi = Module["dynCall_iiddi"] = createExportWrapper("dynCall_iiddi");

/** @type {function(...*:?) *} */
var dynCall_ijjiii = Module["dynCall_ijjiii"] = createExportWrapper("dynCall_ijjiii");

/** @type {function(...*:?) *} */
var dynCall_iisiijii = Module["dynCall_iisiijii"] =
createExportWrapper("dynCall_iisiijii");

/** @type {function(...*:?) *} */
var dynCall_iisiidii = Module["dynCall_iisiidii"] =
createExportWrapper("dynCall_iisiidii");

/** @type {function(...*:?) *} */
var dynCall_iiiffiiiji = Module["dynCall_iiiffiiiji"] =
createExportWrapper("dynCall_iiiffiiiji");

/** @type {function(...*:?) *} */
var dynCall_vijjjii = Module["dynCall_vijjjii"] =
createExportWrapper("dynCall_vijjjii");

/** @type {function(...*:?) *} */
var dynCall_vjiii = Module["dynCall_vjiii"] = createExportWrapper("dynCall_vjiii");

/** @type {function(...*:?) *} */
var dynCall_jiiiiiiii = Module["dynCall_jiiiiiiii"] =
createExportWrapper("dynCall_jiiiiiiii");
```

```
/** @type {function(...*:?) *} */
var dynCall_jjiiiiiiii = Module["dynCall_jjiiiiiiii"] =
createExportWrapper("dynCall_jjiiiiiiii");

/** @type {function(...*:?) *} */
var dynCall_vijfffi = Module["dynCall_vijfffi"] =
createExportWrapper("dynCall_vijfffi");

/** @type {function(...*:?) *} */
var dynCall_iijiiii = Module["dynCall_iijiiii"] =
createExportWrapper("dynCall_iijiiii");

/** @type {function(...*:?) *} */
var dynCall_diiiii = Module["dynCall_diiiii"] = createExportWrapper("dynCall_diiiii");

/** @type {function(...*:?) *} */
var dynCall_viiiffii = Module["dynCall_viiiffii"] =
createExportWrapper("dynCall_viiiffii");

/** @type {function(...*:?) *} */
var dynCall_viiiffi = Module["dynCall_viiiffi"] =
createExportWrapper("dynCall_viiiffi");

/** @type {function(...*:?) *} */
var dynCall_vffffffii = Module["dynCall_vffffffii"] =
createExportWrapper("dynCall_vffffffii");

/** @type {function(...*:?) *} */
var dynCall_jiiiiiiii = Module["dynCall_jiiiiiiii"] =
createExportWrapper("dynCall_jiiiiiiii");

/** @type {function(...*:?) *} */
var dynCall_jidi = Module["dynCall_jidi"] = createExportWrapper("dynCall_jidi");

/** @type {function(...*:?) *} */
var dynCall_vjiiii = Module["dynCall_vjiiii"] = createExportWrapper("dynCall_vjiiii");

/** @type {function(...*:?) *} */
var dynCall_iiiiiiiiiiiii = Module["dynCall_iiiiiiiiiiiii"] =
createExportWrapper("dynCall_iiiiiiiiiiiii");

/** @type {function(...*:?) *} */
var dynCall_viiijji = Module["dynCall_viiijji"] =
createExportWrapper("dynCall_viiijji");

/** @type {function(...*:?) *} */
var dynCall_vijiiiji = Module["dynCall_vijiiiji"] =
createExportWrapper("dynCall_vijiiiji");

/** @type {function(...*:?) *} */
var dynCall_vijiiii = Module["dynCall_vijiiii"] = createExportWrapper("dynCall_vijiiii");

/** @type {function(...*:?) *} */
var dynCall_viiijiji = Module["dynCall_viiijiji"] =
createExportWrapper("dynCall_viiijiji");
```

```
/** @type {function(...*)?:?} */
var dynCall_jjiji = Module["dynCall_jjiji"] = createExportWrapper("dynCall_jjiji");

/** @type {function(...*)?:?} */
var dynCall_ijiji = Module["dynCall_ijiji"] = createExportWrapper("dynCall_ijiji");

/** @type {function(...*)?:?} */
var dynCall_ijijiji = Module["dynCall_ijijiji"] =
createExportWrapper("dynCall_ijijiji");

/** @type {function(...*)?:?} */
var dynCall_ijiiiji = Module["dynCall_ijiiiji"] =
createExportWrapper("dynCall_ijiiiji");

/** @type {function(...*)?:?} */
var dynCall_idiiii = Module["dynCall_idiiii"] = createExportWrapper("dynCall_idiiii");

/** @type {function(...*)?:?} */
var dynCall_idiiiiii = Module["dynCall_idiiiiii"] =
createExportWrapper("dynCall_idiiiiii");

/** @type {function(...*)?:?} */
var dynCall_vjjiiiiii = Module["dynCall_vjjiiiiii"] =
createExportWrapper("dynCall_vjjiiiiii");

/** @type {function(...*)?:?} */
var dynCall_vjjiji = Module["dynCall_vjjiji"] = createExportWrapper("dynCall_vjjiji");

/** @type {function(...*)?:?} */
var dynCall_jijiji = Module["dynCall_jijiji"] = createExportWrapper("dynCall_jijiji");

/** @type {function(...*)?:?} */
var dynCall_jjiiiii = Module["dynCall_jjiiiii"] = createExportWrapper("dynCall_jjiiiii");

/** @type {function(...*)?:?} */
var dynCall_ijiiiiii = Module["dynCall_ijiiiiii"] =
createExportWrapper("dynCall_ijiiiiii");

/** @type {function(...*)?:?} */
var dynCall_fji = Module["dynCall_fji"] = createExportWrapper("dynCall_fji");

/** @type {function(...*)?:?} */
var dynCall_dji = Module["dynCall_dji"] = createExportWrapper("dynCall_dji");

/** @type {function(...*)?:?} */
var dynCall_ddd = Module["dynCall_ddd"] = createExportWrapper("dynCall_ddd");

/** @type {function(...*)?:?} */
var dynCall_vdiiii = Module["dynCall_vdiiii"] = createExportWrapper("dynCall_vdiiii");

/** @type {function(...*)?:?} */
var dynCall_idiiiiii = Module["dynCall_idiiiiii"] = createExportWrapper("dynCall_idiiiiii");

/** @type {function(...*)?:?} */
```

```
var dynCall_jjjii = Module["dynCall_jjjii"] = createExportWrapper("dynCall_jjjii");
/** @type {function(...*)?:?} */
var dynCall_iifiii = Module["dynCall_iifiii"] = createExportWrapper("dynCall_iifiii");
/** @type {function(...*)?:?} */
var dynCall_viiijii = Module["dynCall_viiijii"] =
createExportWrapper("dynCall_viiijii");
/** @type {function(...*)?:?} */
var dynCall_fiff = Module["dynCall_fiff"] = createExportWrapper("dynCall_fiff");
/** @type {function(...*)?:?} */
var dynCall_fif = Module["dynCall_fif"] = createExportWrapper("dynCall_fif");
/** @type {function(...*)?:?} */
var dynCall_jijj = Module["dynCall_jijj"] = createExportWrapper("dynCall_jijj");
/** @type {function(...*)?:?} */
var dynCall_didd = Module["dynCall_didd"] = createExportWrapper("dynCall_didd");
/** @type {function(...*)?:?} */
var dynCall_jij = Module["dynCall_jij"] = createExportWrapper("dynCall_jij");
/** @type {function(...*)?:?} */
var dynCall_did = Module["dynCall_did"] = createExportWrapper("dynCall_did");
/** @type {function(...*)?:?} */
var dynCall_iijjii = Module["dynCall_iijjii"] = createExportWrapper("dynCall_iijjii");
/** @type {function(...*)?:?} */
var dynCall_jijiii = Module["dynCall_jijiii"] = createExportWrapper("dynCall_jijiii");
/** @type {function(...*)?:?} */
var dynCall_vijiisiisi = Module["dynCall_vijiisiisi"] =
createExportWrapper("dynCall_vijiisiisi");
/** @type {function(...*)?:?} */
var dynCall_vijiisiisi = Module["dynCall_vijiisiisi"] =
createExportWrapper("dynCall_vijiisiisi");
/** @type {function(...*)?:?} */
var dynCall_ijjjiijii = Module["dynCall_ijjjiijii"] =
createExportWrapper("dynCall_ijjjiijii");
/** @type {function(...*)?:?} */
var dynCall_iijiiisi = Module["dynCall_iijiiisi"] =
createExportWrapper("dynCall_iijiiisi");
/** @type {function(...*)?:?} */
var dynCall_ijjjiiii = Module["dynCall_ijjjiiii"] =
createExportWrapper("dynCall_ijjjiiii");
/** @type {function(...*)?:?} */
var dynCall_ijjjiiii = Module["dynCall_ijjjiiii"] =
createExportWrapper("dynCall_ijjjiiii");
```

```
var dynCall_vijiiiii = Module["dynCall_vijiiiii"] =
createExportWrapper("dynCall_vijiiiii");

/** @type {function(...*):?} */
var dynCall_ijijiiii = Module["dynCall_ijijiiii"] =
createExportWrapper("dynCall_ijijiiii");

/** @type {function(...*):?} */
var dynCall_iiijjiji = Module["dynCall_iiijjiji"] =
createExportWrapper("dynCall_iiijjiji");

/** @type {function(...*):?} */
var dynCall_vjjjiisi = Module["dynCall_vjjjiisi"] =
createExportWrapper("dynCall_vjjjiisi");

/** @type {function(...*):?} */
var dynCall_jidiii = Module["dynCall_jidiii"] = createExportWrapper("dynCall_jidiii");

/** @type {function(...*):?} */
var dynCall_vijiiiii = Module["dynCall_vijiiiii"] =
createExportWrapper("dynCall_vijiiiii");

/** @type {function(...*):?} */
var dynCall_vjiiisi = Module["dynCall_vjiiisi"] =
createExportWrapper("dynCall_vjiiisi");

/** @type {function(...*):?} */
var dynCall_vijiiisi = Module["dynCall_vijiiisi"] =
createExportWrapper("dynCall_vijiiisi");

/** @type {function(...*):?} */
var dynCall_jjiiisi = Module["dynCall_jjiiisi"] = createExportWrapper("dynCall_jjiiisi");

/** @type {function(...*):?} */
var dynCall_jijjjiji = Module["dynCall_jijjjiji"] =
createExportWrapper("dynCall_jijjjiji");

/** @type {function(...*):?} */
var dynCall_vijjji = Module["dynCall_vijjji"] = createExportWrapper("dynCall_vijjji");

/** @type {function(...*):?} */
var dynCall_ifiiii = Module["dynCall_ifiiii"] = createExportWrapper("dynCall_ifiiii");

/** @type {function(...*):?} */
var dynCall_ifiiisi = Module["dynCall_ifiiisi"] =
createExportWrapper("dynCall_ifiiisi");

/** @type {function(...*):?} */
var dynCall_jfi = Module["dynCall_jfi"] = createExportWrapper("dynCall_jfi");

/** @type {function(...*):?} */
var dynCall_dfi = Module["dynCall_dfi"] = createExportWrapper("dynCall_dfi");
```

```
/** @type {function(...*)?:?} */
var dynCall_jdii = Module["dynCall_jdii"] = createExportWrapper("dynCall_jdii");

/** @type {function(...*)?:?} */
var dynCall_vdii = Module["dynCall_vdii"] = createExportWrapper("dynCall_vdii");

/** @type {function(...*)?:?} */
var dynCall_vijjiiii = Module["dynCall_vijjiiii"] =
createExportWrapper("dynCall_vijjiiii");

/** @type {function(...*)?:?} */
var dynCall_iiiifiiii = Module["dynCall_iiiifiiii"] =
createExportWrapper("dynCall_iiiifiiii");

/** @type {function(...*)?:?} */
var dynCall_vijjii = Module["dynCall_vijjii"] = createExportWrapper("dynCall_vijjii");

/** @type {function(...*)?:?} */
var dynCall_viiiiiiijiji = Module["dynCall_viiiiiiijiji"] =
createExportWrapper("dynCall_viiiiiiijiji");

/** @type {function(...*)?:?} */
var dynCall_viiiiiffii = Module["dynCall_viiiiiffii"] =
createExportWrapper("dynCall_viiiiiffii");

/** @type {function(...*)?:?} */
var dynCall_viiififi = Module["dynCall_viiififi"] =
createExportWrapper("dynCall_viiififi");

/** @type {function(...*)?:?} */
var dynCall_viiififfi = Module["dynCall_viiififfi"] =
createExportWrapper("dynCall_viiififfi");

/** @type {function(...*)?:?} */
var dynCall_iisiifi = Module["dynCall_iisiifi"] =
createExportWrapper("dynCall_iisiifi");

/** @type {function(...*)?:?} */
var dynCall_vififiii = Module["dynCall_vififiii"] =
createExportWrapper("dynCall_vififiii");

/** @type {function(...*)?:?} */
var dynCall_fiffffi = Module["dynCall_fiffffi"] =
createExportWrapper("dynCall_fiffffi");

/** @type {function(...*)?:?} */
var dynCall_iiffiiii = Module["dynCall_iiffiiii"] =
createExportWrapper("dynCall_iiffiiii");

/** @type {function(...*)?:?} */
var dynCall_iiiiffiiii = Module["dynCall_iiiiffiiii"] =
createExportWrapper("dynCall_iiiiffiiii");

/** @type {function(...*)?:?} */
var dynCall_iiffiiii = Module["dynCall_iiffiiii"] =
createExportWrapper("dynCall_iiffiiii");
```

```
var dynCall_viiifiii = Module["dynCall_viiifiii"] =  
createExportWrapper("dynCall_viiifiii");  
  
/** @type {function(...*):?} */  
var dynCall_iifffi = Module["dynCall_iifffi"] = createExportWrapper("dynCall_iifffi");  
  
/** @type {function(...*):?} */  
var dynCall_iidii = Module["dynCall_iidii"] = createExportWrapper("dynCall_iidii");  
  
/** @type {function(...*):?} */  
var dynCall_vijijiji = Module["dynCall_vijijiji"] =  
createExportWrapper("dynCall_vijijiji");  
  
/** @type {function(...*):?} */  
var dynCall_vidii = Module["dynCall_vidii"] = createExportWrapper("dynCall_vidii");  
  
/** @type {function(...*):?} */  
var dynCall_viiiiifii = Module["dynCall_viiiiifii"] =  
createExportWrapper("dynCall_viiiiifii");  
  
/** @type {function(...*):?} */  
var dynCall_ijjjjii = Module["dynCall_ijjjjii"] = createExportWrapper("dynCall_ijjjjii");  
  
/** @type {function(...*):?} */  
var dynCall_ijjjfii = Module["dynCall_ijjjfii"] = createExportWrapper("dynCall_ijjjfii");  
  
/** @type {function(...*):?} */  
var dynCall_ijjjjiji = Module["dynCall_ijjjjiji"] =  
createExportWrapper("dynCall_ijjjjiji");  
  
/** @type {function(...*):?} */  
var dynCall_ijjjiiii = Module["dynCall_ijjjiiii"] =  
createExportWrapper("dynCall_ijjjiiii");  
  
/** @type {function(...*):?} */  
var dynCall_jijjjji = Module["dynCall_jijjjji"] = createExportWrapper("dynCall_jijjjji");  
  
/** @type {function(...*):?} */  
var dynCall_vijjjijiji = Module["dynCall_vijjjijiji"] =  
createExportWrapper("dynCall_vijjjijiji");  
  
/** @type {function(...*):?} */  
var dynCall_ddii = Module["dynCall_ddii"] = createExportWrapper("dynCall_ddii");  
  
/** @type {function(...*):?} */  
var dynCall_iijjji = Module["dynCall_iijjji"] = createExportWrapper("dynCall_iijjji");  
  
/** @type {function(...*):?} */  
var dynCall_vijjjji = Module["dynCall_vijjjji"] =  
createExportWrapper("dynCall_vijjjji");  
  
/** @type {function(...*):?} */  
var dynCall_diddi = Module["dynCall_diddi"] = createExportWrapper("dynCall_diddi");  
  
/** @type {function(...*):?} */
```

```

var dynCall_vij = Module["dynCall_vij"] = createExportWrapper("dynCall_vij");
/** @type {function(...*)?:?} */
var dynCall_iqidiii = Module["dynCall_iqidiii"] =
createExportWrapper("dynCall_iqidiii");

/** @type {function(...*)?:?} */
var dynCall_didii = Module["dynCall_didii"] = createExportWrapper("dynCall_didii");
/** @type {function(...*)?:?} */
var dynCall_vidiji = Module["dynCall_vidiji"] = createExportWrapper("dynCall_vidiji");

/** @type {function(...*)?:?} */
var dynCall_vidjii = Module["dynCall_vidjii"] = createExportWrapper("dynCall_vidjii");
/** @type {function(...*)?:?} */
var dynCall_iidi = Module["dynCall_iidi"] = createExportWrapper("dynCall_iidi");

/** @type {function(...*)?:?} */
var dynCall_vdi = Module["dynCall_vdi"] = createExportWrapper("dynCall_vdi");

/** @type {function(...*)?:?} */
var dynCall_fff = Module["dynCall_fff"] = createExportWrapper("dynCall_fff");
/** @type {function(...*)?:?} */
var dynCall_vif = Module["dynCall_vif"] = createExportWrapper("dynCall_vif");

/** @type {function(...*)?:?} */
var dynCall_viif = Module["dynCall_viif"] = createExportWrapper("dynCall_viif");
/** @type {function(...*)?:?} */
var dynCall_viffff = Module["dynCall_viffff"] = createExportWrapper("dynCall_viffff");

/** @type {function(...*)?:?} */
var dynCall_vid = Module["dynCall_vid"] = createExportWrapper("dynCall_vid");

/** @type {function(...*)?:?} */
var dynCall_viiiiif = Module["dynCall_viiiiif"] =
createExportWrapper("dynCall_viiiiif");

/** @type {function(...*)?:?} */
var dynCall_viiiiif = Module["dynCall_viiiiif"] = createExportWrapper("dynCall_viiiiif");
/** @type {function(...*)?:?} */
var dynCall_viiiiiiif = Module["dynCall_viiiiiiif"] =
createExportWrapper("dynCall_viiiiiiif");

/** @type {function(...*)?:?} */
var dynCall_iijiji = Module["dynCall_iijiji"] =
createExportWrapper("dynCall_iijiji");

/** @type {function(...*)?:?} */
var dynCall_iijij = Module["dynCall_iijij"] = createExportWrapper("dynCall_iijij");

/** @type {function(...*)?:?} */

```

```
var dynCall_iyyyyffff = Module["dynCall_iyyyyffff"] =
createExportWrapper("dynCall_iyyyyffff");

/** @type {function(...*):?} */
var dynCall_iyyyyifiif = Module["dynCall_iyyyyifiif"] =
createExportWrapper("dynCall_iyyyyifiif");

/** @type {function(...*):?} */
var dynCall_iyyyyifiii = Module["dynCall_iyyyyifiii"] =
createExportWrapper("dynCall_iyyyyifiii");

/** @type {function(...*):?} */
var dynCall_iyyyyifiif = Module["dynCall_iyyyyifiif"] =
createExportWrapper("dynCall_iyyyyifiif");

/** @type {function(...*):?} */
var dynCall_fiiiiiiifiifif = Module["dynCall_fiiiiiiifiifif"] =
createExportWrapper("dynCall_fiiiiiiifiifif");

/** @type {function(...*):?} */
var dynCall_fiiiiiiifiiiiif = Module["dynCall_fiiiiiiifiiiiif"] =
createExportWrapper("dynCall_fiiiiiiifiiiiif");

/** @type {function(...*):?} */
var dynCall_vifiiiii = Module["dynCall_vifiiiii"] =
createExportWrapper("dynCall_vifiiiii");

/** @type {function(...*):?} */
var dynCall_iifiiijiji = Module["dynCall_iifiiijiji"] =
createExportWrapper("dynCall_iifiiijiji");

/** @type {function(...*):?} */
var dynCall_vifif = Module["dynCall_vifif"] = createExportWrapper("dynCall_vifif");

/** @type {function(...*):?} */
var dynCall_vifijiji = Module["dynCall_vifijiji"] =
createExportWrapper("dynCall_vifijiji");

/** @type {function(...*):?} */
var dynCall_iiiiffiiii = Module["dynCall_iiiiffiiii"] =
createExportWrapper("dynCall_iiiiffiiii");

/** @type {function(...*):?} */
var dynCall_iiiiffffffi = Module["dynCall_iiiiffffffi"] =
createExportWrapper("dynCall_iiiiffffffi");

/** @type {function(...*):?} */
var dynCall_viffiiiiif = Module["dynCall_viffiiiiif"] =
createExportWrapper("dynCall_viffiiiiif");

/** @type {function(...*):?} */
var dynCall_viffiiffffffiii = Module["dynCall_viffiiffffffiii"] =
createExportWrapper("dynCall_viffiiffffffiii");

/** @type {function(...*):?} */
```

```
var dynCall_viffffiiffiiif = Module["dynCall_viffffiiffiiif"] =
createExportWrapper("dynCall_viffffiiffiiif");

/** @type {function(...*)?:?} */
var dynCall_iiiiffffii = Module["dynCall_iiiiffffii"] =
createExportWrapper("dynCall_iiiiffffii");

/** @type {function(...*)?:?} */
var dynCall_viiiiiiiiifii = Module["dynCall_viiiiiiiiifii"] =
createExportWrapper("dynCall_viiiiiiiiifii");

/** @type {function(...*)?:?} */
var dynCall_viff = Module["dynCall_viff"] = createExportWrapper("dynCall_viff");

/** @type {function(...*)?:?} */
var dynCall_iisiifiiif = Module["dynCall_iisiifiiif"] =
createExportWrapper("dynCall_iisiifiiif");

/** @type {function(...*)?:?} */
var dynCall_iifif = Module["dynCall_iifif"] = createExportWrapper("dynCall_iifif");

/** @type {function(...*)?:?} */
var dynCall_viiff = Module["dynCall_viiff"] = createExportWrapper("dynCall_viiff");

/** @type {function(...*)?:?} */
var dynCall_viiifffi = Module["dynCall_viiifffi"] =
createExportWrapper("dynCall_viiifffi");

/** @type {function(...*)?:?} */
var dynCall_viiifiiiii = Module["dynCall_viiifiiiii"] =
createExportWrapper("dynCall_viiifiiiii");

/** @type {function(...*)?:?} */
var dynCall_viiifiiiiif = Module["dynCall_viiifiiiiif"] =
createExportWrapper("dynCall_viiifiiiiif");

/** @type {function(...*)?:?} */
var dynCall_iiffff = Module["dynCall_iiffff"] = createExportWrapper("dynCall_iiffff");

/** @type {function(...*)?:?} */
var dynCall_iif = Module["dynCall_iif"] = createExportWrapper("dynCall_iif");

/** @type {function(...*)?:?} */
var dynCall_vijj = Module["dynCall_vijj"] = createExportWrapper("dynCall_vijj");

/** @type {function(...*)?:?} */
var dynCall_vijjjj = Module["dynCall_vijjjj"] =
createExportWrapper("dynCall_vijjjj");

/** @type {function(...*)?:?} */
var dynCall_vijjj = Module["dynCall_vijjj"] = createExportWrapper("dynCall_vijjj");

/** @type {function(...*)?:?} */
var dynCall_viiijj = Module["dynCall_viiijj"] = createExportWrapper("dynCall_viiijj");
```

```
/** @type {function(...*:?) *} */
var dynCall_iijiji = Module["dynCall_iijiji"] = createExportWrapper("dynCall_iijiji");

/** @type {function(...*:?) *} */
var dynCall_ijjiiii = Module["dynCall_ijjiiii"] =
createExportWrapper("dynCall_ijjiiii");

/** @type {function(...*:?) *} */
var dynCall_ijj = Module["dynCall_ijj"] = createExportWrapper("dynCall_ijj");

/** @type {function(...*:?) *} */
var dynCall_vjji = Module["dynCall_vjji"] = createExportWrapper("dynCall_vjji");

/** @type {function(...*:?) *} */
var dynCall_iijj = Module["dynCall_iijj"] = createExportWrapper("dynCall_iijj");

/** @type {function(...*:?) *} */
var dynCall_iisiiffiifiii = Module["dynCall_iisiiffiifiii"] =
createExportWrapper("dynCall_iisiiffiifiii");

/** @type {function(...*:?) *} */
var dynCall_viid = Module["dynCall_viid"] = createExportWrapper("dynCall_viid");

/** @type {function(...*:?) *} */
var dynCall_viiif = Module["dynCall_viiif"] = createExportWrapper("dynCall_viiif");

/** @type {function(...*:?) *} */
var dynCall_fiiif = Module["dynCall_fiiif"] = createExportWrapper("dynCall_fiiif");

/** @type {function(...*:?) *} */
var dynCall_iisiiff = Module["dynCall_iisiiff"] =
createExportWrapper("dynCall_iisiiff");

/** @type {function(...*:?) *} */
var dynCall_vjiiiiii = Module["dynCall_vjiiiiii"] =
createExportWrapper("dynCall_vjiiiiii");

/** @type {function(...*:?) *} */
var dynCall_vf = Module["dynCall_vf"] = createExportWrapper("dynCall_vf");

/** @type {function(...*:?) *} */
var dynCall_vffff = Module["dynCall_vffff"] = createExportWrapper("dynCall_vffff");

/** @type {function(...*:?) *} */
var dynCall_vff = Module["dynCall_vff"] = createExportWrapper("dynCall_vff");

/** @type {function(...*:?) *} */
var dynCall_viiiji = Module["dynCall_viiiji"] =
createExportWrapper("dynCall_viiiji");

/** @type {function(...*:?) *} */
var dynCall_viiiiiiiiiiiiii = Module["dynCall_viiiiiiiiiiii"] =
createExportWrapper("dynCall_viiiiiiiiiiii");

/** @type {function(...*:?) *} */
```

```

var dynCall_vifff = Module["dynCall_vifff"] = createExportWrapper("dynCall_vifff");
/** @type {function(...*):?} */
var dynCall_viffff = Module["dynCall_viffff"] = createExportWrapper("dynCall_viffff");
/** @type {function(...*):?} */
var dynCall_ij = Module["dynCall_ij"] = createExportWrapper("dynCall_ij");
/** @type {function(...*):?} */
var dynCall_vfff = Module["dynCall_vfff"] = createExportWrapper("dynCall_vfff");
/** @type {function(...*):?} */
var dynCall_f = Module["dynCall_f"] = createExportWrapper("dynCall_f");
/** @type {function(...*):?} */
var dynCall_vfffffi = Module["dynCall_vfffffi"] =
createExportWrapper("dynCall_vfffffi");
/** @type {function(...*):?} */
var dynCall_ff = Module["dynCall_ff"] = createExportWrapper("dynCall_ff");
/** @type {function(...*):?} */
var dynCall_iiiiiiffiiiiiiiiiffffiii = Module["dynCall_iiiiiiffiiiiiiiiiffffiii"] =
createExportWrapper("dynCall_iiiiiiffiiiiiiiiiffffiii");
/** @type {function(...*):?} */
var dynCall_viififi = Module["dynCall_viififi"] =
createExportWrapper("dynCall_viififi");
/** @type {function(...*):?} */
var dynCall_if = Module["dynCall_if"] = createExportWrapper("dynCall_if");
/** @type {function(...*):?} */
var dynCall_viffiiiiiiii = Module["dynCall_viffiiiiiiii"] =
createExportWrapper("dynCall_viffiiiiiiii");
/** @type {function(...*):?} */
var dynCall_viffiiiiiiii = Module["dynCall_viffiiiiiiii"] =
createExportWrapper("dynCall_viffiiiiiiii");
/** @type {function(...*):?} */
var dynCall_viiiiiiijiii = Module["dynCall_viiiiiiijiii"] =
createExportWrapper("dynCall_viiiiiiijiii");
/** @type {function(...*):?} */
var dynCall_d = Module["dynCall_d"] = createExportWrapper("dynCall_d");
/** @type {function(...*):?} */
var dynCall_vj = Module["dynCall_vj"] = createExportWrapper("dynCall_vj");
/** @type {function(...*):?} */
var dynCall_jiiiiiii = Module["dynCall_jiiiiiii"] =
createExportWrapper("dynCall_jiiiiiii");
/** @type {function(...*):?} */

```

```

var dynCall_jjiiiiiiii = Module["dynCall_jjiiiiiiii"] =
createExportWrapper("dynCall_jjiiiiiiii");

/** @type {function(...*:?) *} */
var dynCall_vijfff = Module["dynCall_vijfff"] = createExportWrapper("dynCall_vijfff");

/** @type {function(...*:?) *} */
var dynCall_ijjji = Module["dynCall_ijjji"] = createExportWrapper("dynCall_ijjji");

/** @type {function(...*:?) *} */
var dynCall_ijjjf = Module["dynCall_ijjjf"] = createExportWrapper("dynCall_ijjjf");

/** @type {function(...*:?) *} */
var dynCall_ijjjji = Module["dynCall_ijjjji"] = createExportWrapper("dynCall_ijjjji");

/** @type {function(...*:?) *} */
var dynCall_ijjjiiii = Module["dynCall_ijjjiiii"] =
createExportWrapper("dynCall_ijjjiiii");

function invoke_i(index,a1) {
    var sp = stackSave();
    try {
        return dynCall_i(index,a1);
    } catch(e) {
        stackRestore(sp);
        if (e !== e+0) throw e;
        _setThrew(1, 0);
    }
}

function invoke_vii(index,a1,a2) {
    var sp = stackSave();
    try {
        dynCall_vii(index,a1,a2);
    } catch(e) {
        stackRestore(sp);
        if (e !== e+0) throw e;
        _setThrew(1, 0);
    }
}

function invoke_v(index) {
    var sp = stackSave();
    try {
        dynCall_v(index);
    } catch(e) {
        stackRestore(sp);
        if (e !== e+0) throw e;
        _setThrew(1, 0);
    }
}

function invoke_iii(index,a1,a2) {
    var sp = stackSave();

```

```

try {
    return dynCall_iii(index,a1,a2);
} catch(e) {
    stackRestore(sp);
    if (e !== e+0) throw e;
    _setThrew(1, 0);
}
}

function invoke_vi(index,a1) {
var sp = stackSave();
try {
    dynCall_vi(index,a1);
} catch(e) {
    stackRestore(sp);
    if (e !== e+0) throw e;
    _setThrew(1, 0);
}
}

function invoke_iiji(index,a1,a2,a3) {
var sp = stackSave();
try {
    return dynCall_iiji(index,a1,a2,a3);
} catch(e) {
    stackRestore(sp);
    if (e !== e+0) throw e;
    _setThrew(1, 0);
}
}

function invoke_iijii(index,a1,a2,a3,a4) {
var sp = stackSave();
try {
    return dynCall_iijii(index,a1,a2,a3,a4);
} catch(e) {
    stackRestore(sp);
    if (e !== e+0) throw e;
    _setThrew(1, 0);
}
}

function invoke_iijiii(index,a1,a2,a3,a4,a5) {
var sp = stackSave();
try {
    return dynCall_iijiii(index,a1,a2,a3,a4,a5);
} catch(e) {
    stackRestore(sp);
    if (e !== e+0) throw e;
    _setThrew(1, 0);
}
}

function invoke_viii(index,a1,a2,a3) {
var sp = stackSave();

```

```

try {
    dynCall_viii(index,a1,a2,a3);
} catch(e) {
    stackRestore(sp);
    if (e !== e+0) throw e;
    _setThrew(1, 0);
}
}

function invoke_i(index) {
var sp = stackSave();
try {
    return dynCall_i(index);
} catch(e) {
    stackRestore(sp);
    if (e !== e+0) throw e;
    _setThrew(1, 0);
}
}

function invoke_viiii(index,a1,a2,a3,a4) {
var sp = stackSave();
try {
    dynCall_viiii(index,a1,a2,a3,a4);
} catch(e) {
    stackRestore(sp);
    if (e !== e+0) throw e;
    _setThrew(1, 0);
}
}

function invoke_iiiiiii(index,a1,a2,a3,a4,a5,a6) {
var sp = stackSave();
try {
    return dynCall_iiiiiii(index,a1,a2,a3,a4,a5,a6);
} catch(e) {
    stackRestore(sp);
    if (e !== e+0) throw e;
    _setThrew(1, 0);
}
}

function invoke_iiiiiiiii(index,a1,a2,a3,a4,a5,a6,a7) {
var sp = stackSave();
try {
    return dynCall_iiiiiiiii(index,a1,a2,a3,a4,a5,a6,a7);
} catch(e) {
    stackRestore(sp);
    if (e !== e+0) throw e;
    _setThrew(1, 0);
}
}

function invoke_iiiiiiiiiii(index,a1,a2,a3,a4,a5,a6,a7,a8,a9,a10) {
var sp = stackSave();

```

```

try {
    return dynCall_iiiiiiiiii(index,a1,a2,a3,a4,a5,a6,a7,a8,a9,a10);
} catch(e) {
    stackRestore(sp);
    if (e !== e+0) throw e;
    _setThrew(1, 0);
}
}

function invoke_iiiiiiiiiii(index,a1,a2,a3,a4,a5,a6,a7,a8,a9,a10,a11,a12) {
var sp = stackSave();
try {
    return dynCall_iiiiiiiiiii(index,a1,a2,a3,a4,a5,a6,a7,a8,a9,a10,a11,a12);
} catch(e) {
    stackRestore(sp);
    if (e !== e+0) throw e;
    _setThrew(1, 0);
}
}

function invoke_fiii(index,a1,a2,a3) {
var sp = stackSave();
try {
    return dynCall_fiii(index,a1,a2,a3);
} catch(e) {
    stackRestore(sp);
    if (e !== e+0) throw e;
    _setThrew(1, 0);
}
}

function invoke_diii(index,a1,a2,a3) {
var sp = stackSave();
try {
    return dynCall_diii(index,a1,a2,a3);
} catch(e) {
    stackRestore(sp);
    if (e !== e+0) throw e;
    _setThrew(1, 0);
}
}

function invoke_viiiiii(index,a1,a2,a3,a4,a5,a6,a7) {
var sp = stackSave();
try {
    dynCall_viiiiii(index,a1,a2,a3,a4,a5,a6,a7);
} catch(e) {
    stackRestore(sp);
    if (e !== e+0) throw e;
    _setThrew(1, 0);
}
}

function invoke_iiiiiiiiii(index,a1,a2,a3,a4,a5,a6,a7,a8,a9,a10,a11) {
var sp = stackSave();

```

```

try {
    return dynCall_iiiiiiiiii(index,a1,a2,a3,a4,a5,a6,a7,a8,a9,a10,a11);
} catch(e) {
    stackRestore(sp);
    if (e !== e+0) throw e;
    _setThrew(1, 0);
}
}

function invoke_viiiiiiiii(index,a1,a2,a3,a4,a5,a6,a7,a8,a9,a10) {
var sp = stackSave();
try {
    dynCall_viiiiiiiii(index,a1,a2,a3,a4,a5,a6,a7,a8,a9,a10);
} catch(e) {
    stackRestore(sp);
    if (e !== e+0) throw e;
    _setThrew(1, 0);
}
}

function invoke_viiiiii(index,a1,a2,a3,a4,a5,a6) {
var sp = stackSave();
try {
    dynCall_viiiiii(index,a1,a2,a3,a4,a5,a6);
} catch(e) {
    stackRestore(sp);
    if (e !== e+0) throw e;
    _setThrew(1, 0);
}
}

function invoke_viiiiiiii(index,a1,a2,a3,a4,a5,a6,a7,a8) {
var sp = stackSave();
try {
    dynCall_viiiiiiii(index,a1,a2,a3,a4,a5,a6,a7,a8);
} catch(e) {
    stackRestore(sp);
    if (e !== e+0) throw e;
    _setThrew(1, 0);
}
}

function invoke_viiiiii(index,a1,a2,a3,a4,a5) {
var sp = stackSave();
try {
    dynCall_viiiiii(index,a1,a2,a3,a4,a5);
} catch(e) {
    stackRestore(sp);
    if (e !== e+0) throw e;
    _setThrew(1, 0);
}
}

function invoke_viiiiiiiiii(index,a1,a2,a3,a4,a5,a6,a7,a8,a9) {
var sp = stackSave();

```

```

try {
    dynCall_viiiiiiiii(index,a1,a2,a3,a4,a5,a6,a7,a8,a9);
} catch(e) {
    stackRestore(sp);
    if (e !== e+0) throw e;
    _setThrew(1, 0);
}
}

function invoke_iiiiiiiiii(index,a1,a2,a3,a4,a5,a6,a7,a8,a9) {
var sp = stackSave();
try {
    return dynCall_iiiiiiiiii(index,a1,a2,a3,a4,a5,a6,a7,a8,a9);
} catch(e) {
    stackRestore(sp);
    if (e !== e+0) throw e;
    _setThrew(1, 0);
}
}

function invoke_viiiiiiiiii(index,a1,a2,a3,a4,a5,a6,a7,a8,a9,a10,a11,a12) {
var sp = stackSave();
try {
    dynCall_viiiiiiiiii(index,a1,a2,a3,a4,a5,a6,a7,a8,a9,a10,a11,a12);
} catch(e) {
    stackRestore(sp);
    if (e !== e+0) throw e;
    _setThrew(1, 0);
}
}

function invoke_fifffi(index,a1,a2,a3,a4) {
var sp = stackSave();
try {
    return dynCall_fifffi(index,a1,a2,a3,a4);
} catch(e) {
    stackRestore(sp);
    if (e !== e+0) throw e;
    _setThrew(1, 0);
}
}

function invoke_iiddi(index,a1,a2,a3,a4) {
var sp = stackSave();
try {
    return dynCall_iiddi(index,a1,a2,a3,a4);
} catch(e) {
    stackRestore(sp);
    if (e !== e+0) throw e;
    _setThrew(1, 0);
}
}

function invoke_iiiiiiiii(index,a1,a2,a3,a4,a5,a6,a7,a8) {
var sp = stackSave();

```

```

try {
    return dynCall_iiiiiiii(index,a1,a2,a3,a4,a5,a6,a7,a8);
} catch(e) {
    stackRestore(sp);
    if (e !== e+0) throw e;
    _setThrew(1, 0);
}
}

function invoke_iividiii(index,a1,a2,a3,a4,a5,a6) {
var sp = stackSave();
try {
    return dynCall_iividiii(index,a1,a2,a3,a4,a5,a6);
} catch(e) {
    stackRestore(sp);
    if (e !== e+0) throw e;
    _setThrew(1, 0);
}
}

function invoke_fiiffi(index,a1,a2,a3,a4,a5) {
var sp = stackSave();
try {
    return dynCall_fiiffi(index,a1,a2,a3,a4,a5);
} catch(e) {
    stackRestore(sp);
    if (e !== e+0) throw e;
    _setThrew(1, 0);
}
}

function invoke_viiififii(index,a1,a2,a3,a4,a5,a6,a7,a8) {
var sp = stackSave();
try {
    dynCall_viiififii(index,a1,a2,a3,a4,a5,a6,a7,a8);
} catch(e) {
    stackRestore(sp);
    if (e !== e+0) throw e;
    _setThrew(1, 0);
}
}

function invoke_vifi(index,a1,a2,a3) {
var sp = stackSave();
try {
    dynCall_vifi(index,a1,a2,a3);
} catch(e) {
    stackRestore(sp);
    if (e !== e+0) throw e;
    _setThrew(1, 0);
}
}

function invoke_vif(index,a1,a2) {
var sp = stackSave();

```

```

try {
    dynCall_vif(index,a1,a2);
} catch(e) {
    stackRestore(sp);
    if (e !== e+0) throw e;
    _setThrew(1, 0);
}
}

function invoke_viffi(index,a1,a2,a3,a4) {
var sp = stackSave();
try {
    dynCall_viffi(index,a1,a2,a3,a4);
} catch(e) {
    stackRestore(sp);
    if (e !== e+0) throw e;
    _setThrew(1, 0);
}
}

function invoke_vifififi(index,a1,a2,a3,a4,a5,a6,a7) {
var sp = stackSave();
try {
    dynCall_vifififi(index,a1,a2,a3,a4,a5,a6,a7);
} catch(e) {
    stackRestore(sp);
    if (e !== e+0) throw e;
    _setThrew(1, 0);
}
}

function invoke_vif(index,a1,a2,a3) {
var sp = stackSave();
try {
    dynCall_vif(index,a1,a2,a3);
} catch(e) {
    stackRestore(sp);
    if (e !== e+0) throw e;
    _setThrew(1, 0);
}
}

function invoke_fi(index,a1) {
var sp = stackSave();
try {
    return dynCall_fi(index,a1);
} catch(e) {
    stackRestore(sp);
    if (e !== e+0) throw e;
    _setThrew(1, 0);
}
}

function invoke_fii(index,a1,a2) {
var sp = stackSave();

```

```

try {
    return dynCall_fii(index,a1,a2);
} catch(e) {
    stackRestore(sp);
    if (e !== e+0) throw e;
    _setThrew(1, 0);
}
}

function invoke_viffii(index,a1,a2,a3,a4,a5) {
var sp = stackSave();
try {
    dynCall_viffii(index,a1,a2,a3,a4,a5);
} catch(e) {
    stackRestore(sp);
    if (e !== e+0) throw e;
    _setThrew(1, 0);
}
}

function invoke_viifi(index,a1,a2,a3,a4) {
var sp = stackSave();
try {
    dynCall_viifi(index,a1,a2,a3,a4);
} catch(e) {
    stackRestore(sp);
    if (e !== e+0) throw e;
    _setThrew(1, 0);
}
}

function invoke_iifi(index,a1,a2,a3) {
var sp = stackSave();
try {
    return dynCall_iifi(index,a1,a2,a3);
} catch(e) {
    stackRestore(sp);
    if (e !== e+0) throw e;
    _setThrew(1, 0);
}
}

function invoke_fifii(index,a1,a2,a3,a4) {
var sp = stackSave();
try {
    return dynCall_fifii(index,a1,a2,a3,a4);
} catch(e) {
    stackRestore(sp);
    if (e !== e+0) throw e;
    _setThrew(1, 0);
}
}

function invoke_vifii(index,a1,a2,a3,a4) {
var sp = stackSave();

```

```

try {
    dynCall_vifii(index,a1,a2,a3,a4);
} catch(e) {
    stackRestore(sp);
    if (e !== e+0) throw e;
    _setThrew(1, 0);
}
}

function invoke_vfiii(index,a1,a2,a3,a4) {
var sp = stackSave();
try {
    dynCall_vfiii(index,a1,a2,a3,a4);
} catch(e) {
    stackRestore(sp);
    if (e !== e+0) throw e;
    _setThrew(1, 0);
}
}

function invoke_vfii(index,a1,a2,a3) {
var sp = stackSave();
try {
    dynCall_vfii(index,a1,a2,a3);
} catch(e) {
    stackRestore(sp);
    if (e !== e+0) throw e;
    _setThrew(1, 0);
}
}

function invoke_ff(index,a1) {
var sp = stackSave();
try {
    return dynCall_ff(index,a1);
} catch(e) {
    stackRestore(sp);
    if (e !== e+0) throw e;
    _setThrew(1, 0);
}
}

function invoke_vfi(index,a1,a2) {
var sp = stackSave();
try {
    dynCall_vfi(index,a1,a2);
} catch(e) {
    stackRestore(sp);
    if (e !== e+0) throw e;
    _setThrew(1, 0);
}
}

function invoke_dddi(index,a1,a2,a3) {
var sp = stackSave();

```

```

try {
    return dynCall_dddi(index,a1,a2,a3);
} catch(e) {
    stackRestore(sp);
    if (e !== e+0) throw e;
    _setThrew(1, 0);
}
}

function invoke_ddi(index,a1,a2) {
var sp = stackSave();
try {
    return dynCall_ddi(index,a1,a2);
} catch(e) {
    stackRestore(sp);
    if (e !== e+0) throw e;
    _setThrew(1, 0);
}
}

function invoke_idi(index,a1,a2) {
var sp = stackSave();
try {
    return dynCall_idi(index,a1,a2);
} catch(e) {
    stackRestore(sp);
    if (e !== e+0) throw e;
    _setThrew(1, 0);
}
}

function invoke_ddiii(index,a1,a2,a3,a4) {
var sp = stackSave();
try {
    return dynCall_ddiii(index,a1,a2,a3,a4);
} catch(e) {
    stackRestore(sp);
    if (e !== e+0) throw e;
    _setThrew(1, 0);
}
}

function invoke_iififi(index,a1,a2,a3,a4,a5) {
var sp = stackSave();
try {
    return dynCall_iififi(index,a1,a2,a3,a4,a5);
} catch(e) {
    stackRestore(sp);
    if (e !== e+0) throw e;
    _setThrew(1, 0);
}
}

function invoke_iiffi(index,a1,a2,a3,a4) {
var sp = stackSave();

```

```

try {
    return dynCall_iiffi(index,a1,a2,a3,a4);
} catch(e) {
    stackRestore(sp);
    if (e !== e+0) throw e;
    _setThrew(1, 0);
}
}

function invoke_iidi(index,a1,a2,a3) {
var sp = stackSave();
try {
    return dynCall_iidi(index,a1,a2,a3);
} catch(e) {
    stackRestore(sp);
    if (e !== e+0) throw e;
    _setThrew(1, 0);
}
}

function invoke_dii(index,a1,a2) {
var sp = stackSave();
try {
    return dynCall_dii(index,a1,a2);
} catch(e) {
    stackRestore(sp);
    if (e !== e+0) throw e;
    _setThrew(1, 0);
}
}

function invoke_viiiiiiiiii(index,a1,a2,a3,a4,a5,a6,a7,a8,a9,a10,a11) {
var sp = stackSave();
try {
    dynCall_viiiiiiiiii(index,a1,a2,a3,a4,a5,a6,a7,a8,a9,a10,a11);
} catch(e) {
    stackRestore(sp);
    if (e !== e+0) throw e;
    _setThrew(1, 0);
}
}

function invoke_viifffiii(index,a1,a2,a3,a4,a5,a6,a7,a8) {
var sp = stackSave();
try {
    dynCall_viifffiii(index,a1,a2,a3,a4,a5,a6,a7,a8);
} catch(e) {
    stackRestore(sp);
    if (e !== e+0) throw e;
    _setThrew(1, 0);
}
}

function invoke_vifiiiiii(index,a1,a2,a3,a4,a5,a6,a7,a8) {
var sp = stackSave();

```

```

try {
    dynCall_vifiiiii(index,a1,a2,a3,a4,a5,a6,a7,a8);
} catch(e) {
    stackRestore(sp);
    if (e !== e+0) throw e;
    _setThrew(1, 0);
}
}

function invoke_vidi(index,a1,a2,a3) {
var sp = stackSave();
try {
    dynCall_vidi(index,a1,a2,a3);
} catch(e) {
    stackRestore(sp);
    if (e !== e+0) throw e;
    _setThrew(1, 0);
}
}

function invoke_iidiidii(index,a1,a2,a3,a4,a5,a6) {
var sp = stackSave();
try {
    return dynCall_iidiidii(index,a1,a2,a3,a4,a5,a6);
} catch(e) {
    stackRestore(sp);
    if (e !== e+0) throw e;
    _setThrew(1, 0);
}
}

function invoke_iidiiii(index,a1,a2,a3,a4,a5) {
var sp = stackSave();
try {
    return dynCall_iidiiii(index,a1,a2,a3,a4,a5);
} catch(e) {
    stackRestore(sp);
    if (e !== e+0) throw e;
    _setThrew(1, 0);
}
}

function invoke_viidi(index,a1,a2,a3,a4) {
var sp = stackSave();
try {
    dynCall_viidi(index,a1,a2,a3,a4);
} catch(e) {
    stackRestore(sp);
    if (e !== e+0) throw e;
    _setThrew(1, 0);
}
}

function invoke_viffffffi(index,a1,a2,a3,a4,a5,a6,a7) {
var sp = stackSave();

```

```

try {
    dynCall_viffffffi(index,a1,a2,a3,a4,a5,a6,a7);
} catch(e) {
    stackRestore(sp);
    if (e !== e+0) throw e;
    _setThrew(1, 0);
}

function invoke_viiifiii(index,a1,a2,a3,a4,a5,a6,a7) {
    var sp = stackSave();
    try {
        dynCall_viiifiii(index,a1,a2,a3,a4,a5,a6,a7);
    } catch(e) {
        stackRestore(sp);
        if (e !== e+0) throw e;
        _setThrew(1, 0);
    }
}

function invoke_fiiiii(index,a1,a2,a3,a4,a5) {
    var sp = stackSave();
    try {
        return dynCall_fiiiii(index,a1,a2,a3,a4,a5);
    } catch(e) {
        stackRestore(sp);
        if (e !== e+0) throw e;
        _setThrew(1, 0);
    }
}

function invoke_fiiiii(index,a1,a2,a3,a4) {
    var sp = stackSave();
    try {
        return dynCall_fiiiii(index,a1,a2,a3,a4);
    } catch(e) {
        stackRestore(sp);
        if (e !== e+0) throw e;
        _setThrew(1, 0);
    }
}

function invoke_viiiiifi(index,a1,a2,a3,a4,a5,a6,a7) {
    var sp = stackSave();
    try {
        dynCall_viiiiifi(index,a1,a2,a3,a4,a5,a6,a7);
    } catch(e) {
        stackRestore(sp);
        if (e !== e+0) throw e;
        _setThrew(1, 0);
    }
}

function invoke_viiiiiffffffiiii(index,a1,a2,a3,a4,a5,a6,a7,a8,a9,a10,a11,a12,a13,a14) {
    var sp = stackSave();
}

```

```

try {
    dynCall_viiiffffffiiiii(index,a1,a2,a3,a4,a5,a6,a7,a8,a9,a10,a11,a12,a13,a14);
} catch(e) {
    stackRestore(sp);
    if (e !== e+0) throw e;
    _setThrew(1, 0);
}
}

function invoke_viffiiiii(index,a1,a2,a3,a4,a5,a6,a7) {
var sp = stackSave();
try {
    dynCall_viffiiiii(index,a1,a2,a3,a4,a5,a6,a7);
} catch(e) {
    stackRestore(sp);
    if (e !== e+0) throw e;
    _setThrew(1, 0);
}
}

function invoke_viiifii(index,a1,a2,a3,a4,a5) {
var sp = stackSave();
try {
    dynCall_viiifii(index,a1,a2,a3,a4,a5);
} catch(e) {
    stackRestore(sp);
    if (e !== e+0) throw e;
    _setThrew(1, 0);
}
}

function invoke_viiiffffffiiii(index,a1,a2,a3,a4,a5,a6,a7,a8,a9,a10,a11) {
var sp = stackSave();
try {
    dynCall_viiiffffffiiii(index,a1,a2,a3,a4,a5,a6,a7,a8,a9,a10,a11);
} catch(e) {
    stackRestore(sp);
    if (e !== e+0) throw e;
    _setThrew(1, 0);
}
}

function invoke_iffi(index,a1,a2,a3) {
var sp = stackSave();
try {
    return dynCall_iffi(index,a1,a2,a3);
} catch(e) {
    stackRestore(sp);
    if (e !== e+0) throw e;
    _setThrew(1, 0);
}
}

function invoke_viff ffi(index,a1,a2,a3,a4,a5,a6) {
var sp = stackSave();

```

```

try {
    dynCall_viffffi(index,a1,a2,a3,a4,a5,a6);
} catch(e) {
    stackRestore(sp);
    if (e !== e+0) throw e;
    _setThrew(1, 0);
}
}

function invoke_vifffi(index,a1,a2,a3,a4,a5) {
var sp = stackSave();
try {
    dynCall_vifffi(index,a1,a2,a3,a4,a5);
} catch(e) {
    stackRestore(sp);
    if (e !== e+0) throw e;
    _setThrew(1, 0);
}
}

function invoke_viiifi(index,a1,a2,a3,a4,a5) {
var sp = stackSave();
try {
    dynCall_viiifi(index,a1,a2,a3,a4,a5);
} catch(e) {
    stackRestore(sp);
    if (e !== e+0) throw e;
    _setThrew(1, 0);
}
}

function invoke_ff ffi(index,a1,a2,a3,a4) {
var sp = stackSave();
try {
    return dynCall_ff ffi(index,a1,a2,a3,a4);
} catch(e) {
    stackRestore(sp);
    if (e !== e+0) throw e;
    _setThrew(1, 0);
}
}

function invoke_viiiffiiiiiiii(index,a1,a2,a3,a4,a5,a6,a7,a8,a9,a10,a11,a12,a13,a14) {
var sp = stackSave();
try {
    dynCall_viiiffiiiiiiii(index,a1,a2,a3,a4,a5,a6,a7,a8,a9,a10,a11,a12,a13,a14);
} catch(e) {
    stackRestore(sp);
    if (e !== e+0) throw e;
    _setThrew(1, 0);
}
}

function invoke_viiiffiiiiii(index,a1,a2,a3,a4,a5,a6,a7,a8,a9,a10,a11) {
var sp = stackSave();

```

```

try {
    dynCall_viiiffiiii(index,a1,a2,a3,a4,a5,a6,a7,a8,a9,a10,a11);
} catch(e) {
    stackRestore(sp);
    if (e !== e+0) throw e;
    _setThrew(1, 0);
}
}

function invoke_viiiiiiiiiiii(index,a1,a2,a3,a4,a5,a6,a7,a8,a9,a10,a11,a12,a13,a14) {
var sp = stackSave();
try {
    dynCall_viiiiiiiiiiii(index,a1,a2,a3,a4,a5,a6,a7,a8,a9,a10,a11,a12,a13,a14);
} catch(e) {
    stackRestore(sp);
    if (e !== e+0) throw e;
    _setThrew(1, 0);
}
}

function invoke_ffii(index,a1,a2,a3) {
var sp = stackSave();
try {
    return dynCall_ffii(index,a1,a2,a3);
} catch(e) {
    stackRestore(sp);
    if (e !== e+0) throw e;
    _setThrew(1, 0);
}
}

function invoke_ffii(index,a1,a2) {
var sp = stackSave();
try {
    return dynCall_ffii(index,a1,a2);
} catch(e) {
    stackRestore(sp);
    if (e !== e+0) throw e;
    _setThrew(1, 0);
}
}

function invoke_viiiffifi(index,a1,a2,a3,a4,a5,a6,a7) {
var sp = stackSave();
try {
    dynCall_viiiffifi(index,a1,a2,a3,a4,a5,a6,a7);
} catch(e) {
    stackRestore(sp);
    if (e !== e+0) throw e;
    _setThrew(1, 0);
}
}

function invoke_viiidi(index,a1,a2,a3,a4,a5) {
var sp = stackSave();

```

```

try {
    dynCall_viiidi(index,a1,a2,a3,a4,a5);
} catch(e) {
    stackRestore(sp);
    if (e !== e+0) throw e;
    _setThrew(1, 0);
}
}

function invoke_fifi(index,a1,a2,a3) {
var sp = stackSave();
try {
    return dynCall_fifi(index,a1,a2,a3);
} catch(e) {
    stackRestore(sp);
    if (e !== e+0) throw e;
    _setThrew(1, 0);
}
}

function invoke_fdi(index,a1,a2) {
var sp = stackSave();
try {
    return dynCall_fdi(index,a1,a2);
} catch(e) {
    stackRestore(sp);
    if (e !== e+0) throw e;
    _setThrew(1, 0);
}
}

function invoke_ifi(index,a1,a2) {
var sp = stackSave();
try {
    return dynCall_ifi(index,a1,a2);
} catch(e) {
    stackRestore(sp);
    if (e !== e+0) throw e;
    _setThrew(1, 0);
}
}

function invoke_viffffii(index,a1,a2,a3,a4,a5,a6,a7) {
var sp = stackSave();
try {
    dynCall_viffffii(index,a1,a2,a3,a4,a5,a6,a7);
} catch(e) {
    stackRestore(sp);
    if (e !== e+0) throw e;
    _setThrew(1, 0);
}
}

function invoke_f(index) {
var sp = stackSave();

```

```

try {
    return dynCall_f(index);
} catch(e) {
    stackRestore(sp);
    if (e !== e+0) throw e;
    _setThrew(1, 0);
}
}

function invoke_iiifi(index,a1,a2,a3,a4,a5) {
var sp = stackSave();
try {
    return dynCall_iiifi(index,a1,a2,a3,a4,a5);
} catch(e) {
    stackRestore(sp);
    if (e !== e+0) throw e;
    _setThrew(1, 0);
}
}

function invoke_fifffi(index,a1,a2,a3,a4,a5) {
var sp = stackSave();
try {
    return dynCall_fifffi(index,a1,a2,a3,a4,a5);
} catch(e) {
    stackRestore(sp);
    if (e !== e+0) throw e;
    _setThrew(1, 0);
}
}

function invoke_iifffi(index,a1,a2,a3,a4,a5) {
var sp = stackSave();
try {
    return dynCall_iifffi(index,a1,a2,a3,a4,a5);
} catch(e) {
    stackRestore(sp);
    if (e !== e+0) throw e;
    _setThrew(1, 0);
}
}

function invoke_ddddi(index,a1,a2,a3,a4) {
var sp = stackSave();
try {
    return dynCall_ddddi(index,a1,a2,a3,a4);
} catch(e) {
    stackRestore(sp);
    if (e !== e+0) throw e;
    _setThrew(1, 0);
}
}

function invoke_fffi(index,a1,a2,a3) {
var sp = stackSave();

```

```

try {
    return dynCall_ffffi(index,a1,a2,a3);
} catch(e) {
    stackRestore(sp);
    if (e !== e+0) throw e;
    _setThrew(1, 0);
}
}

function invoke_iififi(index,a1,a2,a3,a4) {
var sp = stackSave();
try {
    return dynCall_iififi(index,a1,a2,a3,a4);
} catch(e) {
    stackRestore(sp);
    if (e !== e+0) throw e;
    _setThrew(1, 0);
}
}

function invoke_iffffi(index,a1,a2,a3,a4) {
var sp = stackSave();
try {
    return dynCall_iffffi(index,a1,a2,a3,a4);
} catch(e) {
    stackRestore(sp);
    if (e !== e+0) throw e;
    _setThrew(1, 0);
}
}

function invoke_iifiii(index,a1,a2,a3,a4) {
var sp = stackSave();
try {
    return dynCall_iifiii(index,a1,a2,a3,a4);
} catch(e) {
    stackRestore(sp);
    if (e !== e+0) throw e;
    _setThrew(1, 0);
}
}

function invoke_iisiiffiii(index,a1,a2,a3,a4,a5,a6,a7,a8,a9) {
var sp = stackSave();
try {
    return dynCall_iisiiffiii(index,a1,a2,a3,a4,a5,a6,a7,a8,a9);
} catch(e) {
    stackRestore(sp);
    if (e !== e+0) throw e;
    _setThrew(1, 0);
}
}

function invoke_iisiffiiii(index,a1,a2,a3,a4,a5,a6,a7,a8) {
var sp = stackSave();

```

```

try {
    return dynCall_iiiffiiii(index,a1,a2,a3,a4,a5,a6,a7,a8);
} catch(e) {
    stackRestore(sp);
    if (e !== e+0) throw e;
    _setThrew(1, 0);
}
}

function invoke_viiffffi(index,a1,a2,a3,a4,a5,a6,a7) {
var sp = stackSave();
try {
    dynCall_viiffffi(index,a1,a2,a3,a4,a5,a6,a7);
} catch(e) {
    stackRestore(sp);
    if (e !== e+0) throw e;
    _setThrew(1, 0);
}
}

function invoke_viffiii(index,a1,a2,a3,a4,a5,a6) {
var sp = stackSave();
try {
    dynCall_viffiii(index,a1,a2,a3,a4,a5,a6);
} catch(e) {
    stackRestore(sp);
    if (e !== e+0) throw e;
    _setThrew(1, 0);
}
}

function invoke_viffifi(index,a1,a2,a3,a4,a5,a6) {
var sp = stackSave();
try {
    dynCall_viffifi(index,a1,a2,a3,a4,a5,a6);
} catch(e) {
    stackRestore(sp);
    if (e !== e+0) throw e;
    _setThrew(1, 0);
}
}

function invoke_ffffifffi(index,a1,a2,a3,a4,a5,a6,a7) {
var sp = stackSave();
try {
    return dynCall_ffffifffi(index,a1,a2,a3,a4,a5,a6,a7);
} catch(e) {
    stackRestore(sp);
    if (e !== e+0) throw e;
    _setThrew(1, 0);
}
}

function invoke_fffffffi(index,a1,a2,a3,a4,a5,a6,a7) {
var sp = stackSave();

```

```

try {
    return dynCall_fffffff(index,a1,a2,a3,a4,a5,a6,a7);
} catch(e) {
    stackRestore(sp);
    if (e !== e+0) throw e;
    _setThrew(1, 0);
}
}

function invoke_iddi(index,a1,a2,a3) {
var sp = stackSave();
try {
    return dynCall_iddi(index,a1,a2,a3);
} catch(e) {
    stackRestore(sp);
    if (e !== e+0) throw e;
    _setThrew(1, 0);
}
}

function invoke_didi(index,a1,a2,a3) {
var sp = stackSave();
try {
    return dynCall_didi(index,a1,a2,a3);
} catch(e) {
    stackRestore(sp);
    if (e !== e+0) throw e;
    _setThrew(1, 0);
}
}

function invoke_iisiifii(index,a1,a2,a3,a4,a5,a6) {
var sp = stackSave();
try {
    return dynCall_iisiifii(index,a1,a2,a3,a4,a5,a6);
} catch(e) {
    stackRestore(sp);
    if (e !== e+0) throw e;
    _setThrew(1, 0);
}
}

function invoke_viiiffi(index,a1,a2,a3,a4,a5) {
var sp = stackSave();
try {
    dynCall_viiiffi(index,a1,a2,a3,a4,a5);
} catch(e) {
    stackRestore(sp);
    if (e !== e+0) throw e;
    _setThrew(1, 0);
}
}

function invoke_viiifiiii(index,a1,a2,a3,a4,a5,a6,a7) {
var sp = stackSave();

```

```

try {
    dynCall_viiifiii(index,a1,a2,a3,a4,a5,a6,a7);
} catch(e) {
    stackRestore(sp);
    if (e !== e+0) throw e;
    _setThrew(1, 0);
}

function invoke_viiifiii(index,a1,a2,a3,a4,a5,a6) {
    var sp = stackSave();
    try {
        dynCall_viiifiii(index,a1,a2,a3,a4,a5,a6);
    } catch(e) {
        stackRestore(sp);
        if (e !== e+0) throw e;
        _setThrew(1, 0);
    }
}

function invoke_iisiifiii(index,a1,a2,a3,a4,a5,a6,a7) {
    var sp = stackSave();
    try {
        return dynCall_iisiifiii(index,a1,a2,a3,a4,a5,a6,a7);
    } catch(e) {
        stackRestore(sp);
        if (e !== e+0) throw e;
        _setThrew(1, 0);
    }
}

function invoke_iisifiii(index,a1,a2,a3,a4,a5,a6) {
    var sp = stackSave();
    try {
        return dynCall_iisifiii(index,a1,a2,a3,a4,a5,a6);
    } catch(e) {
        stackRestore(sp);
        if (e !== e+0) throw e;
        _setThrew(1, 0);
    }
}

function invoke_iisiifiiii(index,a1,a2,a3,a4,a5,a6,a7,a8) {
    var sp = stackSave();
    try {
        return dynCall_iisiifiiii(index,a1,a2,a3,a4,a5,a6,a7,a8);
    } catch(e) {
        stackRestore(sp);
        if (e !== e+0) throw e;
        _setThrew(1, 0);
    }
}

function invoke_iisifiiii(index,a1,a2,a3,a4,a5,a6,a7) {
    var sp = stackSave();

```

```

try {
    return dynCall_iiifiiii(index,a1,a2,a3,a4,a5,a6,a7);
} catch(e) {
    stackRestore(sp);
    if (e !== e+0) throw e;
    _setThrew(1, 0);
}
}

function
invoke_iiiiiiiffiiiiiiiiiiiiiiii(index,a1,a2,a3,a4,a5,a6,a7,a8,a9,a10,a11,a12,a13,a14,a15,
,a16,a17,a18,a19,a20,a21,a22) {
    var sp = stackSave();
    try {
        return
    }
    dynCall_iiiiiiiffiiiiiiiiiiiiiiii(index,a1,a2,a3,a4,a5,a6,a7,a8,a9,a10,a11,a12,a13,a14,a15,
,a16,a17,a18,a19,a20,a21,a22);
} catch(e) {
    stackRestore(sp);
    if (e !== e+0) throw e;
    _setThrew(1, 0);
}
}

function
invoke_iiiiiiiffiiiiiiiiiffffiiii(index,a1,a2,a3,a4,a5,a6,a7,a8,a9,a10,a11,a12,a13,a14,a15,
,a16,a17,a18,a19,a20,a21,a22,a23,a24) {
    var sp = stackSave();
    try {
        return
    }
    dynCall_iiiiiiiffiiiiiiiiiffffiiii(index,a1,a2,a3,a4,a5,a6,a7,a8,a9,a10,a11,a12,a13,a14,
,a15,a16,a17,a18,a19,a20,a21,a22,a23,a24);
} catch(e) {
    stackRestore(sp);
    if (e !== e+0) throw e;
    _setThrew(1, 0);
}
}

function
invoke_iiiiiiiffiiiiiiiiiffffiiii(index,a1,a2,a3,a4,a5,a6,a7,a8,a9,a10,a11,a12,a13,a14,a15,
,a16,a17,a18,a19,a20,a21,a22,a23) {
    var sp = stackSave();
    try {
        return
    }
    dynCall_iiiiiiiffiiiiiiiiiffffiiii(index,a1,a2,a3,a4,a5,a6,a7,a8,a9,a10,a11,a12,a13,a14,a15,
,a16,a17,a18,a19,a20,a21,a22,a23);
} catch(e) {
    stackRestore(sp);
    if (e !== e+0) throw e;
    _setThrew(1, 0);
}
}

function invoke_viiifiiii(index,a1,a2,a3,a4,a5,a6,a7) {
    var sp = stackSave();
    try {
        dynCall_viiifiiii(index,a1,a2,a3,a4,a5,a6,a7);
    } catch(e) {

```

```
stackRestore(sp);
if (e != e+0) throw e;
```

```

        _setThrew(1, 0);
    }
}

function invoke_fiifi(index,a1,a2,a3,a4) {
    var sp = stackSave();
    try {
        return dynCall_fiifi(index,a1,a2,a3,a4);
    } catch(e) {
        stackRestore(sp);
        if (e !== e+0) throw e;
        _setThrew(1, 0);
    }
}

function invoke_fiifii(index,a1,a2,a3,a4,a5) {
    var sp = stackSave();
    try {
        return dynCall_fiifii(index,a1,a2,a3,a4,a5);
    } catch(e) {
        stackRestore(sp);
        if (e !== e+0) throw e;
        _setThrew(1, 0);
    }
}

function invoke_viiiffi(index,a1,a2,a3,a4,a5,a6) {
    var sp = stackSave();
    try {
        dynCall_viiiffi(index,a1,a2,a3,a4,a5,a6);
    } catch(e) {
        stackRestore(sp);
        if (e !== e+0) throw e;
        _setThrew(1, 0);
    }
}

function invoke_viiifffi(index,a1,a2,a3,a4,a5,a6,a7) {
    var sp = stackSave();
    try {
        dynCall_viiifffi(index,a1,a2,a3,a4,a5,a6,a7);
    } catch(e) {
        stackRestore(sp);
        if (e !== e+0) throw e;
        _setThrew(1, 0);
    }
}

function invoke_viffffi(index,a1,a2,a3,a4,a5,a6) {
    var sp = stackSave();
    try {
        dynCall_viffffi(index,a1,a2,a3,a4,a5,a6);
    } catch(e) {
        stackRestore(sp);
        if (e !== e+0) throw e;
        _setThrew(1, 0);
    }
}

```

```

        _setThrew(1, 0);
    }
}

function invoke_viiifiii(index,a1,a2,a3,a4,a5,a6) {
    var sp = stackSave();
    try {
        dynCall_viiifiii(index,a1,a2,a3,a4,a5,a6);
    } catch(e) {
        stackRestore(sp);
        if (e !== e+0) throw e;
        _setThrew(1, 0);
    }
}

function invoke_viiiiiffi(index,a1,a2,a3,a4,a5,a6,a7,a8) {
    var sp = stackSave();
    try {
        dynCall_viiiiiffi(index,a1,a2,a3,a4,a5,a6,a7,a8);
    } catch(e) {
        stackRestore(sp);
        if (e !== e+0) throw e;
        _setThrew(1, 0);
    }
}

function invoke_viiiiiiiiii(index,a1,a2,a3,a4,a5,a6,a7,a8,a9,a10,a11,a12,a13) {
    var sp = stackSave();
    try {
        dynCall_viiiiiiiiii(index,a1,a2,a3,a4,a5,a6,a7,a8,a9,a10,a11,a12,a13);
    } catch(e) {
        stackRestore(sp);
        if (e !== e+0) throw e;
        _setThrew(1, 0);
    }
}

function invoke_viiifi(index,a1,a2,a3,a4,a5,a6) {
    var sp = stackSave();
    try {
        dynCall_viiifi(index,a1,a2,a3,a4,a5,a6);
    } catch(e) {
        stackRestore(sp);
        if (e !== e+0) throw e;
        _setThrew(1, 0);
    }
}

function invoke_iisiiffiiii(index,a1,a2,a3,a4,a5,a6,a7,a8,a9,a10) {
    var sp = stackSave();
    try {
        return dynCall_iisiiffiiii(index,a1,a2,a3,a4,a5,a6,a7,a8,a9,a10);
    } catch(e) {
        stackRestore(sp);
        if (e !== e+0) throw e;
    }
}

```

```

        _setThrew(1, 0);
    }
}

function invoke_fiiiffi(index,a1,a2,a3,a4,a5,a6) {
    var sp = stackSave();
    try {
        return dynCall_fiiiffi(index,a1,a2,a3,a4,a5,a6);
    } catch(e) {
        stackRestore(sp);
        if (e !== e+0) throw e;
        _setThrew(1, 0);
    }
}

function invoke_vffi(index,a1,a2,a3) {
    var sp = stackSave();
    try {
        dynCall_vffi(index,a1,a2,a3);
    } catch(e) {
        stackRestore(sp);
        if (e !== e+0) throw e;
        _setThrew(1, 0);
    }
}

function invoke_fiiiifi(index,a1,a2,a3,a4,a5,a6) {
    var sp = stackSave();
    try {
        return dynCall_fiiiifi(index,a1,a2,a3,a4,a5,a6);
    } catch(e) {
        stackRestore(sp);
        if (e !== e+0) throw e;
        _setThrew(1, 0);
    }
}

function invoke_diiidi(index,a1,a2,a3,a4,a5,a6) {
    var sp = stackSave();
    try {
        return dynCall_diiidi(index,a1,a2,a3,a4,a5,a6);
    } catch(e) {
        stackRestore(sp);
        if (e !== e+0) throw e;
        _setThrew(1, 0);
    }
}

function invoke_if(index,a1) {
    var sp = stackSave();
    try {
        return dynCall_if(index,a1);
    } catch(e) {
        stackRestore(sp);
        if (e !== e+0) throw e;
    }
}

```

```

        _setThrew(1, 0);
    }
}

function invoke_viffiiiiiiii(index,a1,a2,a3,a4,a5,a6,a7,a8,a9,a10,a11,a12,a13,a14) {
    var sp = stackSave();
    try {
        dynCall_viffiiiiiiii(index,a1,a2,a3,a4,a5,a6,a7,a8,a9,a10,a11,a12,a13,a14);
    } catch(e) {
        stackRestore(sp);
        if (e !== e+0) throw e;
        _setThrew(1, 0);
    }
}

function invoke_viffiiiiiiii(index,a1,a2,a3,a4,a5,a6,a7,a8,a9,a10,a11,a12,a13) {
    var sp = stackSave();
    try {
        dynCall_viffiiiiiiii(index,a1,a2,a3,a4,a5,a6,a7,a8,a9,a10,a11,a12,a13);
    } catch(e) {
        stackRestore(sp);
        if (e !== e+0) throw e;
        _setThrew(1, 0);
    }
}

function invoke_viffiiiiii(index,a1,a2,a3,a4,a5,a6,a7,a8,a9,a10,a11) {
    var sp = stackSave();
    try {
        dynCall_viffiiiiii(index,a1,a2,a3,a4,a5,a6,a7,a8,a9,a10,a11);
    } catch(e) {
        stackRestore(sp);
        if (e !== e+0) throw e;
        _setThrew(1, 0);
    }
}

function invoke_viffiiiiii(index,a1,a2,a3,a4,a5,a6,a7,a8,a9,a10) {
    var sp = stackSave();
    try {
        dynCall_viffiiiiii(index,a1,a2,a3,a4,a5,a6,a7,a8,a9,a10);
    } catch(e) {
        stackRestore(sp);
        if (e !== e+0) throw e;
        _setThrew(1, 0);
    }
}

function invoke_viff(index,a1,a2,a3,a4) {
    var sp = stackSave();
    try {
        dynCall_viff(index,a1,a2,a3,a4);
    } catch(e) {
        stackRestore(sp);
        if (e !== e+0) throw e;
    }
}

```

```

        _setThrew(1, 0);
    }
}

function invoke_iisiifiiiii(index,a1,a2,a3,a4,a5,a6,a7,a8,a9,a10) {
    var sp = stackSave();
    try {
        return dynCall_iisiifiiiii(index,a1,a2,a3,a4,a5,a6,a7,a8,a9,a10);
    } catch(e) {
        stackRestore(sp);
        if (e !== e+0) throw e;
        _setThrew(1, 0);
    }
}

function invoke_iisiifiiiii(index,a1,a2,a3,a4,a5,a6,a7,a8,a9) {
    var sp = stackSave();
    try {
        return dynCall_iisiifiiiii(index,a1,a2,a3,a4,a5,a6,a7,a8,a9);
    } catch(e) {
        stackRestore(sp);
        if (e !== e+0) throw e;
        _setThrew(1, 0);
    }
}

function invoke_iisiifiiiii(index,a1,a2,a3,a4,a5,a6,a7,a8) {
    var sp = stackSave();
    try {
        return dynCall_iisiifiiiii(index,a1,a2,a3,a4,a5,a6,a7,a8);
    } catch(e) {
        stackRestore(sp);
        if (e !== e+0) throw e;
        _setThrew(1, 0);
    }
}

function invoke_di(index,a1) {
    var sp = stackSave();
    try {
        return dynCall_di(index,a1);
    } catch(e) {
        stackRestore(sp);
        if (e !== e+0) throw e;
        _setThrew(1, 0);
    }
}

function invoke_viiidi(index,a1,a2,a3,a4,a5,a6) {
    var sp = stackSave();
    try {
        dynCall_viiidi(index,a1,a2,a3,a4,a5,a6);
    } catch(e) {
        stackRestore(sp);
        if (e !== e+0) throw e;
    }
}

```

```

        _setThrew(1, 0);
    }
}

function invoke_iiiiiiidii(index,a1,a2,a3,a4,a5,a6,a7,a8,a9) {
    var sp = stackSave();
    try {
        return dynCall_iiiiiiidii(index,a1,a2,a3,a4,a5,a6,a7,a8,a9);
    } catch(e) {
        stackRestore(sp);
        if (e !== e+0) throw e;
        _setThrew(1, 0);
    }
}

function invoke_viidii(index,a1,a2,a3,a4,a5) {
    var sp = stackSave();
    try {
        dynCall_viidii(index,a1,a2,a3,a4,a5);
    } catch(e) {
        stackRestore(sp);
        if (e !== e+0) throw e;
        _setThrew(1, 0);
    }
}

function invoke_fiiiiiii(index,a1,a2,a3,a4,a5,a6,a7) {
    var sp = stackSave();
    try {
        return dynCall_fiiiiiii(index,a1,a2,a3,a4,a5,a6,a7);
    } catch(e) {
        stackRestore(sp);
        if (e !== e+0) throw e;
        _setThrew(1, 0);
    }
}

function invoke_d(index) {
    var sp = stackSave();
    try {
        return dynCall_d(index);
    } catch(e) {
        stackRestore(sp);
        if (e !== e+0) throw e;
        _setThrew(1, 0);
    }
}

function invoke_vf(index,a1) {
    var sp = stackSave();
    try {
        dynCall_vf(index,a1);
    } catch(e) {
        stackRestore(sp);
        if (e !== e+0) throw e;
    }
}

```

```

        _setThrew(1, 0);
    }
}

function invoke_iqidii(index,a1,a2,a3,a4,a5) {
    var sp = stackSave();
    try {
        return dynCall_iqidii(index,a1,a2,a3,a4,a5);
    } catch(e) {
        stackRestore(sp);
        if (e !== e+0) throw e;
        _setThrew(1, 0);
    }
}

function invoke_fffffi(index,a1,a2,a3,a4,a5) {
    var sp = stackSave();
    try {
        return dynCall_fffffi(index,a1,a2,a3,a4,a5);
    } catch(e) {
        stackRestore(sp);
        if (e !== e+0) throw e;
        _setThrew(1, 0);
    }
}

function invoke_didii(index,a1,a2,a3,a4) {
    var sp = stackSave();
    try {
        return dynCall_didii(index,a1,a2,a3,a4);
    } catch(e) {
        stackRestore(sp);
        if (e !== e+0) throw e;
        _setThrew(1, 0);
    }
}

function invoke_vfffi(index,a1,a2,a3,a4) {
    var sp = stackSave();
    try {
        dynCall_vfffi(index,a1,a2,a3,a4);
    } catch(e) {
        stackRestore(sp);
        if (e !== e+0) throw e;
        _setThrew(1, 0);
    }
}

function invoke_viffffiii(index,a1,a2,a3,a4,a5,a6,a7,a8) {
    var sp = stackSave();
    try {
        dynCall_viffffiii(index,a1,a2,a3,a4,a5,a6,a7,a8);
    } catch(e) {
        stackRestore(sp);
        if (e !== e+0) throw e;
    }
}

```

```

        _setThrew(1, 0);
    }
}

function invoke_viffffffi(index,a1,a2,a3,a4,a5,a6,a7,a8) {
    var sp = stackSave();
    try {
        dynCall_viffffffi(index,a1,a2,a3,a4,a5,a6,a7,a8);
    } catch(e) {
        stackRestore(sp);
        if (e !== e+0) throw e;
        _setThrew(1, 0);
    }
}

function invoke_viiiiifiiiii(index,a1,a2,a3,a4,a5,a6,a7,a8,a9,a10,a11,a12,a13) {
    var sp = stackSave();
    try {
        dynCall_viiiiifiiiii(index,a1,a2,a3,a4,a5,a6,a7,a8,a9,a10,a11,a12,a13);
    } catch(e) {
        stackRestore(sp);
        if (e !== e+0) throw e;
        _setThrew(1, 0);
    }
}

function invoke_vfffffi(index,a1,a2,a3,a4,a5) {
    var sp = stackSave();
    try {
        dynCall_vfffffi(index,a1,a2,a3,a4,a5);
    } catch(e) {
        stackRestore(sp);
        if (e !== e+0) throw e;
        _setThrew(1, 0);
    }
}

function invoke_diidi(index,a1,a2,a3,a4) {
    var sp = stackSave();
    try {
        return dynCall_diidi(index,a1,a2,a3,a4);
    } catch(e) {
        stackRestore(sp);
        if (e !== e+0) throw e;
        _setThrew(1, 0);
    }
}

function invoke_viiidii(index,a1,a2,a3,a4,a5,a6,a7) {
    var sp = stackSave();
    try {
        dynCall_viiidii(index,a1,a2,a3,a4,a5,a6,a7);
    } catch(e) {
        stackRestore(sp);
        if (e !== e+0) throw e;

```

```

        _setThrew(1, 0);
    }
}

function invoke_vidiiiii(index,a1,a2,a3,a4,a5,a6,a7) {
    var sp = stackSave();
    try {
        dynCall_vidiiiii(index,a1,a2,a3,a4,a5,a6,a7);
    } catch(e) {
        stackRestore(sp);
        if (e !== e+0) throw e;
        _setThrew(1, 0);
    }
}

function invoke_diiiii(index,a1,a2,a3,a4) {
    var sp = stackSave();
    try {
        return dynCall_diiiii(index,a1,a2,a3,a4);
    } catch(e) {
        stackRestore(sp);
        if (e !== e+0) throw e;
        _setThrew(1, 0);
    }
}

function invoke_vidii(index,a1,a2,a3,a4) {
    var sp = stackSave();
    try {
        dynCall_vidii(index,a1,a2,a3,a4);
    } catch(e) {
        stackRestore(sp);
        if (e !== e+0) throw e;
        _setThrew(1, 0);
    }
}

function invoke_iisiidii(index,a1,a2,a3,a4,a5,a6,a7) {
    var sp = stackSave();
    try {
        return dynCall_iisiidii(index,a1,a2,a3,a4,a5,a6,a7);
    } catch(e) {
        stackRestore(sp);
        if (e !== e+0) throw e;
        _setThrew(1, 0);
    }
}

function invoke_diiiiii(index,a1,a2,a3,a4,a5) {
    var sp = stackSave();
    try {
        return dynCall_diiiiii(index,a1,a2,a3,a4,a5);
    } catch(e) {
        stackRestore(sp);
        if (e !== e+0) throw e;
    }
}

```

```

        _setThrew(1, 0);
    }
}

function invoke_vfff(index,a1,a2,a3) {
    var sp = stackSave();
    try {
        dynCall_vfff(index,a1,a2,a3);
    } catch(e) {
        stackRestore(sp);
        if (e !== e+0) throw e;
        _setThrew(1, 0);
    }
}

function invoke_vffff(index,a1,a2,a3,a4) {
    var sp = stackSave();
    try {
        dynCall_vffff(index,a1,a2,a3,a4);
    } catch(e) {
        stackRestore(sp);
        if (e !== e+0) throw e;
        _setThrew(1, 0);
    }
}

function invoke_viiif(index,a1,a2,a3,a4) {
    var sp = stackSave();
    try {
        dynCall_viiif(index,a1,a2,a3,a4);
    } catch(e) {
        stackRestore(sp);
        if (e !== e+0) throw e;
        _setThrew(1, 0);
    }
}

function invoke_viiiffi(index,a1,a2,a3,a4,a5,a6,a7) {
    var sp = stackSave();
    try {
        dynCall_viiiffi(index,a1,a2,a3,a4,a5,a6,a7);
    } catch(e) {
        stackRestore(sp);
        if (e !== e+0) throw e;
        _setThrew(1, 0);
    }
}

function invoke_viiiiffi(index,a1,a2,a3,a4,a5,a6,a7) {
    var sp = stackSave();
    try {
        dynCall_viiiiffi(index,a1,a2,a3,a4,a5,a6,a7);
    } catch(e) {
        stackRestore(sp);
        if (e !== e+0) throw e;
    }
}

```

```

        _setThrew(1, 0);
    }
}

function invoke_vfffffffii(index,a1,a2,a3,a4,a5,a6,a7,a8) {
    var sp = stackSave();
    try {
        dynCall_vfffffffii(index,a1,a2,a3,a4,a5,a6,a7,a8);
    } catch(e) {
        stackRestore(sp);
        if (e !== e+0) throw e;
        _setThrew(1, 0);
    }
}

function invoke_vfffffffii(index,a1,a2,a3,a4,a5,a6,a7) {
    var sp = stackSave();
    try {
        dynCall_vfffffffii(index,a1,a2,a3,a4,a5,a6,a7);
    } catch(e) {
        stackRestore(sp);
        if (e !== e+0) throw e;
        _setThrew(1, 0);
    }
}

function invoke_iiiiiiiiiiiii(index,a1,a2,a3,a4,a5,a6,a7,a8,a9,a10,a11,a12,a13) {
    var sp = stackSave();
    try {
        return dynCall_iiiiiiiiiiiii(index,a1,a2,a3,a4,a5,a6,a7,a8,a9,a10,a11,a12,a13);
    } catch(e) {
        stackRestore(sp);
        if (e !== e+0) throw e;
        _setThrew(1, 0);
    }
}

function invoke_idiiii(index,a1,a2,a3,a4) {
    var sp = stackSave();
    try {
        return dynCall_idiiii(index,a1,a2,a3,a4);
    } catch(e) {
        stackRestore(sp);
        if (e !== e+0) throw e;
        _setThrew(1, 0);
    }
}

function invoke_idiiiii(index,a1,a2,a3,a4,a5,a6) {
    var sp = stackSave();
    try {
        return dynCall_idiiiii(index,a1,a2,a3,a4,a5,a6);
    } catch(e) {
        stackRestore(sp);
        if (e !== e+0) throw e;

```

```

        _setThrew(1, 0);
    }
}

function invoke_ddd(index,a1,a2) {
    var sp = stackSave();
    try {
        return dynCall_ddd(index,a1,a2);
    } catch(e) {
        stackRestore(sp);
        if (e !== e+0) throw e;
        _setThrew(1, 0);
    }
}

function invoke_vdiii(index,a1,a2,a3,a4) {
    var sp = stackSave();
    try {
        dynCall_vdiii(index,a1,a2,a3,a4);
    } catch(e) {
        stackRestore(sp);
        if (e !== e+0) throw e;
        _setThrew(1, 0);
    }
}

function invoke_idiiii(index,a1,a2,a3,a4,a5) {
    var sp = stackSave();
    try {
        return dynCall_idiiii(index,a1,a2,a3,a4,a5);
    } catch(e) {
        stackRestore(sp);
        if (e !== e+0) throw e;
        _setThrew(1, 0);
    }
}

function invoke_iifiiii(index,a1,a2,a3,a4,a5) {
    var sp = stackSave();
    try {
        return dynCall_iifiiii(index,a1,a2,a3,a4,a5);
    } catch(e) {
        stackRestore(sp);
        if (e !== e+0) throw e;
        _setThrew(1, 0);
    }
}

function invoke_iisiiffiiii(index,a1,a2,a3,a4,a5,a6,a7,a8,a9) {
    var sp = stackSave();
    try {
        return dynCall_iisiiffiiii(index,a1,a2,a3,a4,a5,a6,a7,a8,a9);
    } catch(e) {
        stackRestore(sp);
        if (e !== e+0) throw e;
    }
}

```

```
        _setThrew(1, 0);
    }
}

function invoke_fiff(index,a1,a2,a3) {
    var sp = stackSave();
    try {
        return dynCall_fiff(index,a1,a2,a3);
    } catch(e) {
        stackRestore(sp);
        if (e !== e+0) throw e;
        _setThrew(1, 0);
    }
}

function invoke_fif(index,a1,a2) {
    var sp = stackSave();
    try {
        return dynCall_fif(index,a1,a2);
    } catch(e) {
        stackRestore(sp);
        if (e !== e+0) throw e;
        _setThrew(1, 0);
    }
}

function invoke_didd(index,a1,a2,a3) {
    var sp = stackSave();
    try {
        return dynCall_didd(index,a1,a2,a3);
    } catch(e) {
        stackRestore(sp);
        if (e !== e+0) throw e;
        _setThrew(1, 0);
    }
}

function invoke_did(index,a1,a2) {
    var sp = stackSave();
    try {
        return dynCall_did(index,a1,a2);
    } catch(e) {
        stackRestore(sp);
        if (e !== e+0) throw e;
        _setThrew(1, 0);
    }
}

function invoke_ifiiii(index,a1,a2,a3,a4) {
    var sp = stackSave();
    try {
        return dynCall_ifiiii(index,a1,a2,a3,a4);
    } catch(e) {
        stackRestore(sp);
        if (e !== e+0) throw e;
    }
}
```

```

        _setThrew(1, 0);
    }
}

function invoke_ifiiiii(index,a1,a2,a3,a4,a5,a6) {
    var sp = stackSave();
    try {
        return dynCall_ifiiiii(index,a1,a2,a3,a4,a5,a6);
    } catch(e) {
        stackRestore(sp);
        if (e !== e+0) throw e;
        _setThrew(1, 0);
    }
}

function invoke_dfi(index,a1,a2) {
    var sp = stackSave();
    try {
        return dynCall_dfi(index,a1,a2);
    } catch(e) {
        stackRestore(sp);
        if (e !== e+0) throw e;
        _setThrew(1, 0);
    }
}

function invoke_vdii(index,a1,a2,a3) {
    var sp = stackSave();
    try {
        dynCall_vdii(index,a1,a2,a3);
    } catch(e) {
        stackRestore(sp);
        if (e !== e+0) throw e;
        _setThrew(1, 0);
    }
}

function invoke_iij(index,a1,a2,a3) {
    var sp = stackSave();
    try {
        return dynCall_iij(index,a1,a2,a3);
    } catch(e) {
        stackRestore(sp);
        if (e !== e+0) throw e;
        _setThrew(1, 0);
    }
}

function invoke_iiijiii(index,a1,a2,a3,a4,a5,a6,a7) {
    var sp = stackSave();
    try {
        return dynCall_iiijiii(index,a1,a2,a3,a4,a5,a6,a7);
    } catch(e) {
        stackRestore(sp);
        if (e !== e+0) throw e;
    }
}

```

```

        _setThrew(1, 0);
    }
}

function invoke_jiiii(index,a1,a2,a3,a4) {
    var sp = stackSave();
    try {
        return dynCall_jiiii(index,a1,a2,a3,a4);
    } catch(e) {
        stackRestore(sp);
        if (e !== e+0) throw e;
        _setThrew(1, 0);
    }
}

function invoke_ji(index,a1) {
    var sp = stackSave();
    try {
        return dynCall_ji(index,a1);
    } catch(e) {
        stackRestore(sp);
        if (e !== e+0) throw e;
        _setThrew(1, 0);
    }
}

function invoke_jii(index,a1,a2) {
    var sp = stackSave();
    try {
        return dynCall_jii(index,a1,a2);
    } catch(e) {
        stackRestore(sp);
        if (e !== e+0) throw e;
        _setThrew(1, 0);
    }
}

function invoke_vijiji(index,a1,a2,a3,a4,a5,a6) {
    var sp = stackSave();
    try {
        dynCall_vijiji(index,a1,a2,a3,a4,a5,a6);
    } catch(e) {
        stackRestore(sp);
        if (e !== e+0) throw e;
        _setThrew(1, 0);
    }
}

function invoke_j(index) {
    var sp = stackSave();
    try {
        return dynCall_j(index);
    } catch(e) {
        stackRestore(sp);
        if (e !== e+0) throw e;

```

```

        _setThrew(1, 0);
    }
}

function invoke_ijji(index,a1,a2,a3,a4,a5) {
    var sp = stackSave();
    try {
        return dynCall_ijji(index,a1,a2,a3,a4,a5);
    } catch(e) {
        stackRestore(sp);
        if (e !== e+0) throw e;
        _setThrew(1, 0);
    }
}

function invoke_vijiiiiii(index,a1,a2,a3,a4,a5,a6,a7,a8,a9,a10) {
    var sp = stackSave();
    try {
        dynCall_vijiiiiii(index,a1,a2,a3,a4,a5,a6,a7,a8,a9,a10);
    } catch(e) {
        stackRestore(sp);
        if (e !== e+0) throw e;
        _setThrew(1, 0);
    }
}

function invoke_vijiii(index,a1,a2,a3,a4,a5,a6) {
    var sp = stackSave();
    try {
        dynCall_vijiii(index,a1,a2,a3,a4,a5,a6);
    } catch(e) {
        stackRestore(sp);
        if (e !== e+0) throw e;
        _setThrew(1, 0);
    }
}

function invoke_jji(index,a1,a2,a3) {
    var sp = stackSave();
    try {
        return dynCall_jji(index,a1,a2,a3);
    } catch(e) {
        stackRestore(sp);
        if (e !== e+0) throw e;
        _setThrew(1, 0);
    }
}

function invoke_viji(index,a1,a2,a3,a4) {
    var sp = stackSave();
    try {
        dynCall_viji(index,a1,a2,a3,a4);
    } catch(e) {
        stackRestore(sp);
        if (e !== e+0) throw e;

```

```

        _setThrew(1, 0);
    }
}

function invoke_ijiji(index,a1,a2,a3) {
    var sp = stackSave();
    try {
        return dynCall_ijiji(index,a1,a2,a3);
    } catch(e) {
        stackRestore(sp);
        if (e !== e+0) throw e;
        _setThrew(1, 0);
    }
}

function invoke_jjjji(index,a1,a2,a3,a4,a5) {
    var sp = stackSave();
    try {
        return dynCall_jjjji(index,a1,a2,a3,a4,a5);
    } catch(e) {
        stackRestore(sp);
        if (e !== e+0) throw e;
        _setThrew(1, 0);
    }
}

function invoke_vijiji(index,a1,a2,a3,a4,a5) {
    var sp = stackSave();
    try {
        dynCall_vijiji(index,a1,a2,a3,a4,a5);
    } catch(e) {
        stackRestore(sp);
        if (e !== e+0) throw e;
        _setThrew(1, 0);
    }
}

function invoke_jjiji(index,a1,a2,a3,a4) {
    var sp = stackSave();
    try {
        return dynCall_jjiji(index,a1,a2,a3,a4);
    } catch(e) {
        stackRestore(sp);
        if (e !== e+0) throw e;
        _setThrew(1, 0);
    }
}

function invoke_iiji(index,a1,a2,a3,a4) {
    var sp = stackSave();
    try {
        return dynCall_iiji(index,a1,a2,a3,a4);
    } catch(e) {
        stackRestore(sp);
        if (e !== e+0) throw e;

```

```

        _setThrew(1, 0);
    }
}

function invoke_vijji(index,a1,a2,a3,a4,a5,a6) {
    var sp = stackSave();
    try {
        dynCall_vijji(index,a1,a2,a3,a4,a5,a6);
    } catch(e) {
        stackRestore(sp);
        if (e !== e+0) throw e;
        _setThrew(1, 0);
    }
}

function invoke_jiiiiiiiii(index,a1,a2,a3,a4,a5,a6,a7,a8,a9,a10) {
    var sp = stackSave();
    try {
        return dynCall_jiiiiiiiii(index,a1,a2,a3,a4,a5,a6,a7,a8,a9,a10);
    } catch(e) {
        stackRestore(sp);
        if (e !== e+0) throw e;
        _setThrew(1, 0);
    }
}

function invoke_iijiji(index,a1,a2,a3,a4,a5) {
    var sp = stackSave();
    try {
        return dynCall_iijiji(index,a1,a2,a3,a4,a5);
    } catch(e) {
        stackRestore(sp);
        if (e !== e+0) throw e;
        _setThrew(1, 0);
    }
}

function invoke_jiiiii(index,a1,a2,a3,a4,a5) {
    var sp = stackSave();
    try {
        return dynCall_jiiiii(index,a1,a2,a3,a4,a5);
    } catch(e) {
        stackRestore(sp);
        if (e !== e+0) throw e;
        _setThrew(1, 0);
    }
}

function invoke_jiii(index,a1,a2,a3) {
    var sp = stackSave();
    try {
        return dynCall_jiii(index,a1,a2,a3);
    } catch(e) {
        stackRestore(sp);
        if (e !== e+0) throw e;

```

```

        _setThrew(1, 0);
    }
}

function invoke_vji(index,a1,a2,a3) {
    var sp = stackSave();
    try {
        dynCall_vji(index,a1,a2,a3);
    } catch(e) {
        stackRestore(sp);
        if (e !== e+0) throw e;
        _setThrew(1, 0);
    }
}

function invoke_iijiji(index,a1,a2,a3,a4,a5,a6) {
    var sp = stackSave();
    try {
        return dynCall_iijiji(index,a1,a2,a3,a4,a5,a6);
    } catch(e) {
        stackRestore(sp);
        if (e !== e+0) throw e;
        _setThrew(1, 0);
    }
}

function invoke_iiiji(index,a1,a2,a3,a4,a5) {
    var sp = stackSave();
    try {
        return dynCall_iiiji(index,a1,a2,a3,a4,a5);
    } catch(e) {
        stackRestore(sp);
        if (e !== e+0) throw e;
        _setThrew(1, 0);
    }
}

function invoke_ijiiiii(index,a1,a2,a3,a4,a5,a6) {
    var sp = stackSave();
    try {
        return dynCall_ijiiiii(index,a1,a2,a3,a4,a5,a6);
    } catch(e) {
        stackRestore(sp);
        if (e !== e+0) throw e;
        _setThrew(1, 0);
    }
}

function invoke_ijiji(index,a1,a2,a3,a4,a5) {
    var sp = stackSave();
    try {
        return dynCall_ijiji(index,a1,a2,a3,a4,a5);
    } catch(e) {
        stackRestore(sp);
        if (e !== e+0) throw e;
        _setThrew(1, 0);
    }
}

```

```

        _setThrew(1, 0);
    }
}

function invoke_viji(index,a1,a2,a3,a4,a5) {
    var sp = stackSave();
    try {
        dynCall_viji(index,a1,a2,a3,a4,a5);
    } catch(e) {
        stackRestore(sp);
        if (e !== e+0) throw e;
        _setThrew(1, 0);
    }
}

function invoke_iiji(index,a1,a2,a3,a4,a5,a6,a7) {
    var sp = stackSave();
    try {
        return dynCall_iiji(index,a1,a2,a3,a4,a5,a6,a7);
    } catch(e) {
        stackRestore(sp);
        if (e !== e+0) throw e;
        _setThrew(1, 0);
    }
}

function invoke_vijiji(index,a1,a2,a3,a4,a5,a6,a7) {
    var sp = stackSave();
    try {
        dynCall_vijiji(index,a1,a2,a3,a4,a5,a6,a7);
    } catch(e) {
        stackRestore(sp);
        if (e !== e+0) throw e;
        _setThrew(1, 0);
    }
}

function invoke_vijijiji(index,a1,a2,a3,a4,a5,a6,a7,a8,a9) {
    var sp = stackSave();
    try {
        dynCall_vijijiji(index,a1,a2,a3,a4,a5,a6,a7,a8,a9);
    } catch(e) {
        stackRestore(sp);
        if (e !== e+0) throw e;
        _setThrew(1, 0);
    }
}

function invoke_vijijiji(index,a1,a2,a3,a4,a5,a6,a7,a8,a9,a10) {
    var sp = stackSave();
    try {
        dynCall_vijijiji(index,a1,a2,a3,a4,a5,a6,a7,a8,a9,a10);
    } catch(e) {
        stackRestore(sp);
        if (e !== e+0) throw e;
    }
}

```

```

        _setThrew(1, 0);
    }
}

function invoke_jiiiii(index,a1,a2,a3,a4,a5,a6) {
    var sp = stackSave();
    try {
        return dynCall_jiiiii(index,a1,a2,a3,a4,a5,a6);
    } catch(e) {
        stackRestore(sp);
        if (e !== e+0) throw e;
        _setThrew(1, 0);
    }
}

function invoke_jiji(index,a1,a2,a3,a4) {
    var sp = stackSave();
    try {
        return dynCall_jiji(index,a1,a2,a3,a4);
    } catch(e) {
        stackRestore(sp);
        if (e !== e+0) throw e;
        _setThrew(1, 0);
    }
}

function invoke_iijiii(index,a1,a2,a3,a4,a5,a6) {
    var sp = stackSave();
    try {
        return dynCall_iijiii(index,a1,a2,a3,a4,a5,a6);
    } catch(e) {
        stackRestore(sp);
        if (e !== e+0) throw e;
        _setThrew(1, 0);
    }
}

function invoke_jdi(index,a1,a2) {
    var sp = stackSave();
    try {
        return dynCall_jdi(index,a1,a2);
    } catch(e) {
        stackRestore(sp);
        if (e !== e+0) throw e;
        _setThrew(1, 0);
    }
}

function invoke_viiiji(index,a1,a2,a3,a4,a5,a6) {
    var sp = stackSave();
    try {
        dynCall_viiiji(index,a1,a2,a3,a4,a5,a6);
    } catch(e) {
        stackRestore(sp);
        if (e !== e+0) throw e;
    }
}

```

```

        _setThrew(1, 0);
    }
}

function invoke_iiiiiiiiji(index,a1,a2,a3,a4,a5,a6,a7,a8,a9,a10,a11) {
    var sp = stackSave();
    try {
        return dynCall_iiiiiiiiji(index,a1,a2,a3,a4,a5,a6,a7,a8,a9,a10,a11);
    } catch(e) {
        stackRestore(sp);
        if (e !== e+0) throw e;
        _setThrew(1, 0);
    }
}

function invoke_viijiji(index,a1,a2,a3,a4,a5,a6,a7) {
    var sp = stackSave();
    try {
        dynCall_viijiji(index,a1,a2,a3,a4,a5,a6,a7);
    } catch(e) {
        stackRestore(sp);
        if (e !== e+0) throw e;
        _setThrew(1, 0);
    }
}

function invoke_vijiji(index,a1,a2,a3,a4,a5,a6,a7) {
    var sp = stackSave();
    try {
        dynCall_vijiji(index,a1,a2,a3,a4,a5,a6,a7);
    } catch(e) {
        stackRestore(sp);
        if (e !== e+0) throw e;
        _setThrew(1, 0);
    }
}

function invoke_jjjiji(index,a1,a2,a3,a4,a5,a6,a7) {
    var sp = stackSave();
    try {
        return dynCall_jjjiji(index,a1,a2,a3,a4,a5,a6,a7);
    } catch(e) {
        stackRestore(sp);
        if (e !== e+0) throw e;
        _setThrew(1, 0);
    }
}

function invoke_jijiji(index,a1,a2,a3,a4,a5,a6) {
    var sp = stackSave();
    try {
        return dynCall_jijiji(index,a1,a2,a3,a4,a5,a6);
    } catch(e) {
        stackRestore(sp);
        if (e !== e+0) throw e;

```

```

        _setThrew(1, 0);
    }
}

function invoke_ijiji(index,a1,a2,a3,a4) {
    var sp = stackSave();
    try {
        return dynCall_ijiji(index,a1,a2,a3,a4);
    } catch(e) {
        stackRestore(sp);
        if (e !== e+0) throw e;
        _setThrew(1, 0);
    }
}

function invoke_jiiiji(index,a1,a2,a3,a4,a5,a6,a7) {
    var sp = stackSave();
    try {
        return dynCall_jiiiji(index,a1,a2,a3,a4,a5,a6,a7);
    } catch(e) {
        stackRestore(sp);
        if (e !== e+0) throw e;
        _setThrew(1, 0);
    }
}

function invoke_viidiji(index,a1,a2,a3,a4,a5,a6,a7) {
    var sp = stackSave();
    try {
        dynCall_viidiji(index,a1,a2,a3,a4,a5,a6,a7);
    } catch(e) {
        stackRestore(sp);
        if (e !== e+0) throw e;
        _setThrew(1, 0);
    }
}

function invoke_viidjii(index,a1,a2,a3,a4,a5,a6,a7) {
    var sp = stackSave();
    try {
        dynCall_viidjii(index,a1,a2,a3,a4,a5,a6,a7);
    } catch(e) {
        stackRestore(sp);
        if (e !== e+0) throw e;
        _setThrew(1, 0);
    }
}

function invoke_jijiji(index,a1,a2,a3,a4,a5) {
    var sp = stackSave();
    try {
        return dynCall_jijiji(index,a1,a2,a3,a4,a5);
    } catch(e) {
        stackRestore(sp);
        if (e !== e+0) throw e;

```

```

        _setThrew(1, 0);
    }
}

function invoke_vijiji(index,a1,a2,a3,a4,a5,a6,a7,a8,a9) {
    var sp = stackSave();
    try {
        dynCall_vijiji(index,a1,a2,a3,a4,a5,a6,a7,a8,a9);
    } catch(e) {
        stackRestore(sp);
        if (e !== e+0) throw e;
        _setThrew(1, 0);
    }
}

function invoke_viidjii(index,a1,a2,a3,a4,a5,a6,a7,a8) {
    var sp = stackSave();
    try {
        dynCall_viidjii(index,a1,a2,a3,a4,a5,a6,a7,a8);
    } catch(e) {
        stackRestore(sp);
        if (e !== e+0) throw e;
        _setThrew(1, 0);
    }
}

function invoke_ij(index,a1,a2) {
    var sp = stackSave();
    try {
        return dynCall_ij(index,a1,a2);
    } catch(e) {
        stackRestore(sp);
        if (e !== e+0) throw e;
        _setThrew(1, 0);
    }
}

function invoke_vjii(index,a1,a2,a3,a4) {
    var sp = stackSave();
    try {
        dynCall_vjii(index,a1,a2,a3,a4);
    } catch(e) {
        stackRestore(sp);
        if (e !== e+0) throw e;
        _setThrew(1, 0);
    }
}

function invoke_viiiiiiijiii(index,a1,a2,a3,a4,a5,a6,a7,a8,a9,a10,a11,a12,a13,a14) {
    var sp = stackSave();
    try {
        dynCall_viiiiiiijiii(index,a1,a2,a3,a4,a5,a6,a7,a8,a9,a10,a11,a12,a13,a14);
    } catch(e) {
        stackRestore(sp);
        if (e !== e+0) throw e;
    }
}

```

```

        _setThrew(1, 0);
    }
}

function invoke_viiiiiiijiii(index,a1,a2,a3,a4,a5,a6,a7,a8,a9,a10,a11,a12,a13) {
    var sp = stackSave();
    try {
        dynCall_viiiiiiijiii(index,a1,a2,a3,a4,a5,a6,a7,a8,a9,a10,a11,a12,a13);
    } catch(e) {
        stackRestore(sp);
        if (e !== e+0) throw e;
        _setThrew(1, 0);
    }
}

function invoke_jiji(index,a1,a2,a3,a4,a5) {
    var sp = stackSave();
    try {
        return dynCall_jiji(index,a1,a2,a3,a4,a5);
    } catch(e) {
        stackRestore(sp);
        if (e !== e+0) throw e;
        _setThrew(1, 0);
    }
}

function invoke_iijiji(index,a1,a2,a3,a4,a5,a6) {
    var sp = stackSave();
    try {
        return dynCall_iijiji(index,a1,a2,a3,a4,a5,a6);
    } catch(e) {
        stackRestore(sp);
        if (e !== e+0) throw e;
        _setThrew(1, 0);
    }
}

function invoke_ijjiji(index,a1,a2,a3,a4,a5,a6,a7) {
    var sp = stackSave();
    try {
        return dynCall_ijjiji(index,a1,a2,a3,a4,a5,a6,a7);
    } catch(e) {
        stackRestore(sp);
        if (e !== e+0) throw e;
        _setThrew(1, 0);
    }
}

function invoke_iisiijii(index,a1,a2,a3,a4,a5,a6,a7,a8) {
    var sp = stackSave();
    try {
        return dynCall_iisiijii(index,a1,a2,a3,a4,a5,a6,a7,a8);
    } catch(e) {
        stackRestore(sp);
        if (e !== e+0) throw e;
        _setThrew(1, 0);
    }
}

```

```

        _setThrew(1, 0);
    }
}

function invoke_iisiiffiiiji(index,a1,a2,a3,a4,a5,a6,a7,a8,a9,a10,a11) {
    var sp = stackSave();
    try {
        return dynCall_iisiiffiiiji(index,a1,a2,a3,a4,a5,a6,a7,a8,a9,a10,a11);
    } catch(e) {
        stackRestore(sp);
        if (e !== e+0) throw e;
        _setThrew(1, 0);
    }
}

function invoke_vijjjii(index,a1,a2,a3,a4,a5,a6,a7,a8) {
    var sp = stackSave();
    try {
        dynCall_vijjjii(index,a1,a2,a3,a4,a5,a6,a7,a8);
    } catch(e) {
        stackRestore(sp);
        if (e !== e+0) throw e;
        _setThrew(1, 0);
    }
}

function invoke_vj(index,a1,a2) {
    var sp = stackSave();
    try {
        dynCall_vj(index,a1,a2);
    } catch(e) {
        stackRestore(sp);
        if (e !== e+0) throw e;
        _setThrew(1, 0);
    }
}

function invoke_vjiii(index,a1,a2,a3,a4,a5) {
    var sp = stackSave();
    try {
        dynCall_vjiii(index,a1,a2,a3,a4,a5);
    } catch(e) {
        stackRestore(sp);
        if (e !== e+0) throw e;
        _setThrew(1, 0);
    }
}

function invoke_jiiiiiiii(index,a1,a2,a3,a4,a5,a6,a7,a8) {
    var sp = stackSave();
    try {
        return dynCall_jiiiiiiii(index,a1,a2,a3,a4,a5,a6,a7,a8);
    } catch(e) {
        stackRestore(sp);
        if (e !== e+0) throw e;
    }
}

```

```

        _setThrew(1, 0);
    }
}

function invoke_jjiiiiiiii(index,a1,a2,a3,a4,a5,a6,a7,a8,a9,a10,a11,a12) {
    var sp = stackSave();
    try {
        return dynCall_jjiiiiiiii(index,a1,a2,a3,a4,a5,a6,a7,a8,a9,a10,a11,a12);
    } catch(e) {
        stackRestore(sp);
        if (e !== e+0) throw e;
        _setThrew(1, 0);
    }
}

function invoke_jiiiiiii(index,a1,a2,a3,a4,a5,a6,a7) {
    var sp = stackSave();
    try {
        return dynCall_jiiiiiii(index,a1,a2,a3,a4,a5,a6,a7);
    } catch(e) {
        stackRestore(sp);
        if (e !== e+0) throw e;
        _setThrew(1, 0);
    }
}

function invoke_jjiiiiiiii(index,a1,a2,a3,a4,a5,a6,a7,a8,a9,a10,a11) {
    var sp = stackSave();
    try {
        return dynCall_jjiiiiiiii(index,a1,a2,a3,a4,a5,a6,a7,a8,a9,a10,a11);
    } catch(e) {
        stackRestore(sp);
        if (e !== e+0) throw e;
        _setThrew(1, 0);
    }
}

function invoke_vijfffi(index,a1,a2,a3,a4,a5,a6,a7) {
    var sp = stackSave();
    try {
        dynCall_vijfffi(index,a1,a2,a3,a4,a5,a6,a7);
    } catch(e) {
        stackRestore(sp);
        if (e !== e+0) throw e;
        _setThrew(1, 0);
    }
}

function invoke_vijfff(index,a1,a2,a3,a4,a5,a6) {
    var sp = stackSave();
    try {
        dynCall_vijfff(index,a1,a2,a3,a4,a5,a6);
    } catch(e) {
        stackRestore(sp);
        if (e !== e+0) throw e;
        _setThrew(1, 0);
    }
}

```

```

        _setThrew(1, 0);
    }
}

function invoke_vij(index,a1,a2,a3) {
    var sp = stackSave();
    try {
        dynCall_vij(index,a1,a2,a3);
    } catch(e) {
        stackRestore(sp);
        if (e !== e+0) throw e;
        _setThrew(1, 0);
    }
}

function invoke_iijiiii(index,a1,a2,a3,a4,a5,a6,a7) {
    var sp = stackSave();
    try {
        return dynCall_iijiiii(index,a1,a2,a3,a4,a5,a6,a7);
    } catch(e) {
        stackRestore(sp);
        if (e !== e+0) throw e;
        _setThrew(1, 0);
    }
}

function invoke_viji(index,a1,a2,a3,a4) {
    var sp = stackSave();
    try {
        dynCall_viji(index,a1,a2,a3,a4);
    } catch(e) {
        stackRestore(sp);
        if (e !== e+0) throw e;
        _setThrew(1, 0);
    }
}

function invoke_iiji(index,a1,a2,a3,a4) {
    var sp = stackSave();
    try {
        return dynCall_iiji(index,a1,a2,a3,a4);
    } catch(e) {
        stackRestore(sp);
        if (e !== e+0) throw e;
        _setThrew(1, 0);
    }
}

function invoke_jidi(index,a1,a2,a3) {
    var sp = stackSave();
    try {
        return dynCall_jidi(index,a1,a2,a3);
    } catch(e) {
        stackRestore(sp);
        if (e !== e+0) throw e;
    }
}

```

```

        _setThrew(1, 0);
    }
}

function invoke_vjiiiii(index,a1,a2,a3,a4,a5,a6) {
    var sp = stackSave();
    try {
        dynCall_vjiiiii(index,a1,a2,a3,a4,a5,a6);
    } catch(e) {
        stackRestore(sp);
        if (e !== e+0) throw e;
        _setThrew(1, 0);
    }
}

function invoke_viiijji(index,a1,a2,a3,a4,a5,a6,a7,a8) {
    var sp = stackSave();
    try {
        dynCall_viiijji(index,a1,a2,a3,a4,a5,a6,a7,a8);
    } catch(e) {
        stackRestore(sp);
        if (e !== e+0) throw e;
        _setThrew(1, 0);
    }
}

function invoke_vijiijiji(index,a1,a2,a3,a4,a5,a6,a7,a8,a9,a10,a11) {
    var sp = stackSave();
    try {
        dynCall_vijiijiji(index,a1,a2,a3,a4,a5,a6,a7,a8,a9,a10,a11);
    } catch(e) {
        stackRestore(sp);
        if (e !== e+0) throw e;
        _setThrew(1, 0);
    }
}

function invoke_viiijiji(index,a1,a2,a3,a4,a5,a6,a7,a8,a9,a10) {
    var sp = stackSave();
    try {
        dynCall_viiijiji(index,a1,a2,a3,a4,a5,a6,a7,a8,a9,a10);
    } catch(e) {
        stackRestore(sp);
        if (e !== e+0) throw e;
        _setThrew(1, 0);
    }
}

function invoke_ijiji(index,a1,a2,a3,a4,a5,a6,a7) {
    var sp = stackSave();
    try {
        return dynCall_ijiji(index,a1,a2,a3,a4,a5,a6,a7);
    } catch(e) {
        stackRestore(sp);
        if (e !== e+0) throw e;
    }
}

```

```

        _setThrew(1, 0);
    }
}

function invoke_ijiiiji(index,a1,a2,a3,a4,a5,a6,a7,a8,a9) {
    var sp = stackSave();
    try {
        return dynCall_ijiiiji(index,a1,a2,a3,a4,a5,a6,a7,a8,a9);
    } catch(e) {
        stackRestore(sp);
        if (e !== e+0) throw e;
        _setThrew(1, 0);
    }
}

function invoke_vjjiiiii(index,a1,a2,a3,a4,a5,a6,a7,a8,a9) {
    var sp = stackSave();
    try {
        dynCall_vjjiiiii(index,a1,a2,a3,a4,a5,a6,a7,a8,a9);
    } catch(e) {
        stackRestore(sp);
        if (e !== e+0) throw e;
        _setThrew(1, 0);
    }
}

function invoke_vjjiji(index,a1,a2,a3,a4,a5,a6) {
    var sp = stackSave();
    try {
        dynCall_vjjiji(index,a1,a2,a3,a4,a5,a6);
    } catch(e) {
        stackRestore(sp);
        if (e !== e+0) throw e;
        _setThrew(1, 0);
    }
}

function invoke_ijijiji(index,a1,a2,a3,a4,a5,a6,a7,a8) {
    var sp = stackSave();
    try {
        return dynCall_ijijiji(index,a1,a2,a3,a4,a5,a6,a7,a8);
    } catch(e) {
        stackRestore(sp);
        if (e !== e+0) throw e;
        _setThrew(1, 0);
    }
}

function invoke_jjiiiii(index,a1,a2,a3,a4,a5,a6) {
    var sp = stackSave();
    try {
        return dynCall_jjiiiii(index,a1,a2,a3,a4,a5,a6);
    } catch(e) {
        stackRestore(sp);
        if (e !== e+0) throw e;
    }
}

```

```

        _setThrew(1, 0);
    }
}

function invoke_ijiiiii(index,a1,a2,a3,a4,a5,a6,a7) {
    var sp = stackSave();
    try {
        return dynCall_ijiiiii(index,a1,a2,a3,a4,a5,a6,a7);
    } catch(e) {
        stackRestore(sp);
        if (e !== e+0) throw e;
        _setThrew(1, 0);
    }
}

function invoke_fji(index,a1,a2,a3) {
    var sp = stackSave();
    try {
        return dynCall_fji(index,a1,a2,a3);
    } catch(e) {
        stackRestore(sp);
        if (e !== e+0) throw e;
        _setThrew(1, 0);
    }
}

function invoke_dji(index,a1,a2,a3) {
    var sp = stackSave();
    try {
        return dynCall_dji(index,a1,a2,a3);
    } catch(e) {
        stackRestore(sp);
        if (e !== e+0) throw e;
        _setThrew(1, 0);
    }
}

function invoke_jjjiii(index,a1,a2,a3,a4,a5,a6) {
    var sp = stackSave();
    try {
        return dynCall_jjjiii(index,a1,a2,a3,a4,a5,a6);
    } catch(e) {
        stackRestore(sp);
        if (e !== e+0) throw e;
        _setThrew(1, 0);
    }
}

function invoke_viiijiji(index,a1,a2,a3,a4,a5,a6,a7,a8) {
    var sp = stackSave();
    try {
        dynCall_viiijiji(index,a1,a2,a3,a4,a5,a6,a7,a8);
    } catch(e) {
        stackRestore(sp);
        if (e !== e+0) throw e;
    }
}

```

```

        _setThrew(1, 0);
    }
}

function invoke_jijj(index,a1,a2,a3,a4,a5) {
    var sp = stackSave();
    try {
        return dynCall_jijj(index,a1,a2,a3,a4,a5);
    } catch(e) {
        stackRestore(sp);
        if (e !== e+0) throw e;
        _setThrew(1, 0);
    }
}

function invoke_jij(index,a1,a2,a3) {
    var sp = stackSave();
    try {
        return dynCall_jij(index,a1,a2,a3);
    } catch(e) {
        stackRestore(sp);
        if (e !== e+0) throw e;
        _setThrew(1, 0);
    }
}

function invoke_iijjji(index,a1,a2,a3,a4,a5,a6,a7) {
    var sp = stackSave();
    try {
        return dynCall_iijjji(index,a1,a2,a3,a4,a5,a6,a7);
    } catch(e) {
        stackRestore(sp);
        if (e !== e+0) throw e;
        _setThrew(1, 0);
    }
}

function invoke_jijihi(index,a1,a2,a3,a4,a5,a6) {
    var sp = stackSave();
    try {
        return dynCall_jijihi(index,a1,a2,a3,a4,a5,a6);
    } catch(e) {
        stackRestore(sp);
        if (e !== e+0) throw e;
        _setThrew(1, 0);
    }
}

function invoke_vijiiiiiii(index,a1,a2,a3,a4,a5,a6,a7,a8,a9,a10,a11) {
    var sp = stackSave();
    try {
        dynCall_vijiiiiiii(index,a1,a2,a3,a4,a5,a6,a7,a8,a9,a10,a11);
    } catch(e) {
        stackRestore(sp);
        if (e !== e+0) throw e;
    }
}

```

```

        _setThrew(1, 0);
    }
}

function invoke_iijiiiiii(index,a1,a2,a3,a4,a5,a6,a7,a8,a9) {
    var sp = stackSave();
    try {
        return dynCall_iijiiiiii(index,a1,a2,a3,a4,a5,a6,a7,a8,a9);
    } catch(e) {
        stackRestore(sp);
        if (e !== e+0) throw e;
        _setThrew(1, 0);
    }
}

function invoke_ijijiiiii(index,a1,a2,a3,a4,a5,a6,a7,a8,a9,a10) {
    var sp = stackSave();
    try {
        return dynCall_ijijiiiii(index,a1,a2,a3,a4,a5,a6,a7,a8,a9,a10);
    } catch(e) {
        stackRestore(sp);
        if (e !== e+0) throw e;
        _setThrew(1, 0);
    }
}

function invoke_ijjjijiji(index,a1,a2,a3,a4,a5,a6,a7,a8,a9,a10,a11,a12) {
    var sp = stackSave();
    try {
        return dynCall_ijjjijiji(index,a1,a2,a3,a4,a5,a6,a7,a8,a9,a10,a11,a12);
    } catch(e) {
        stackRestore(sp);
        if (e !== e+0) throw e;
        _setThrew(1, 0);
    }
}

function invoke_vjjjiiiii(index,a1,a2,a3,a4,a5,a6,a7,a8,a9,a10) {
    var sp = stackSave();
    try {
        dynCall_vjjjiiiii(index,a1,a2,a3,a4,a5,a6,a7,a8,a9,a10);
    } catch(e) {
        stackRestore(sp);
        if (e !== e+0) throw e;
        _setThrew(1, 0);
    }
}

function invoke_jidii(index,a1,a2,a3,a4) {
    var sp = stackSave();
    try {
        return dynCall_jidii(index,a1,a2,a3,a4);
    } catch(e) {
        stackRestore(sp);
        if (e !== e+0) throw e;

```

```

        _setThrew(1, 0);
    }
}

function invoke_vijiyyyy(index,a1,a2,a3,a4,a5,a6,a7,a8,a9) {
    var sp = stackSave();
    try {
        dynCall_vijiyyyy(index,a1,a2,a3,a4,a5,a6,a7,a8,a9);
    } catch(e) {
        stackRestore(sp);
        if (e !== e+0) throw e;
        _setThrew(1, 0);
    }
}

function invoke_vijiyyy(index,a1,a2,a3,a4,a5,a6,a7) {
    var sp = stackSave();
    try {
        dynCall_vijiyyy(index,a1,a2,a3,a4,a5,a6,a7);
    } catch(e) {
        stackRestore(sp);
        if (e !== e+0) throw e;
        _setThrew(1, 0);
    }
}

function invoke_jjiiii(index,a1,a2,a3,a4,a5) {
    var sp = stackSave();
    try {
        return dynCall_jjiiii(index,a1,a2,a3,a4,a5);
    } catch(e) {
        stackRestore(sp);
        if (e !== e+0) throw e;
        _setThrew(1, 0);
    }
}

function invoke_jjiiiiii(index,a1,a2,a3,a4,a5,a6,a7) {
    var sp = stackSave();
    try {
        return dynCall_jjiiiiii(index,a1,a2,a3,a4,a5,a6,a7);
    } catch(e) {
        stackRestore(sp);
        if (e !== e+0) throw e;
        _setThrew(1, 0);
    }
}

function invoke_jijjjjii(index,a1,a2,a3,a4,a5,a6,a7,a8,a9) {
    var sp = stackSave();
    try {
        return dynCall_jijjjjii(index,a1,a2,a3,a4,a5,a6,a7,a8,a9);
    } catch(e) {
        stackRestore(sp);
        if (e !== e+0) throw e;
        _setThrew(1, 0);
    }
}

```

```

        _setThrew(1, 0);
    }
}

function invoke_vijjji(index,a1,a2,a3,a4,a5,a6,a7,a8) {
    var sp = stackSave();
    try {
        dynCall_vijjji(index,a1,a2,a3,a4,a5,a6,a7,a8);
    } catch(e) {
        stackRestore(sp);
        if (e !== e+0) throw e;
        _setThrew(1, 0);
    }
}

function invoke_ijjjiii(index,a1,a2,a3,a4,a5,a6,a7,a8,a9) {
    var sp = stackSave();
    try {
        return dynCall_ijjjiii(index,a1,a2,a3,a4,a5,a6,a7,a8,a9);
    } catch(e) {
        stackRestore(sp);
        if (e !== e+0) throw e;
        _setThrew(1, 0);
    }
}

function invoke_vijiellijii(index,a1,a2,a3,a4,a5,a6,a7,a8,a9,a10) {
    var sp = stackSave();
    try {
        dynCall_vijiellijii(index,a1,a2,a3,a4,a5,a6,a7,a8,a9,a10);
    } catch(e) {
        stackRestore(sp);
        if (e !== e+0) throw e;
        _setThrew(1, 0);
    }
}

function invoke_iijjjiji(index,a1,a2,a3,a4,a5,a6,a7,a8,a9) {
    var sp = stackSave();
    try {
        return dynCall_iijjjiji(index,a1,a2,a3,a4,a5,a6,a7,a8,a9);
    } catch(e) {
        stackRestore(sp);
        if (e !== e+0) throw e;
        _setThrew(1, 0);
    }
}

function invoke_vjiiiii(index,a1,a2,a3,a4,a5,a6,a7) {
    var sp = stackSave();
    try {
        dynCall_vjiiiii(index,a1,a2,a3,a4,a5,a6,a7);
    } catch(e) {
        stackRestore(sp);
        if (e !== e+0) throw e;

```

```

        _setThrew(1, 0);
    }
}

function invoke_jfi(index,a1,a2) {
    var sp = stackSave();
    try {
        return dynCall_jfi(index,a1,a2);
    } catch(e) {
        stackRestore(sp);
        if (e !== e+0) throw e;
        _setThrew(1, 0);
    }
}

function invoke_jdii(index,a1,a2,a3) {
    var sp = stackSave();
    try {
        return dynCall_jdii(index,a1,a2,a3);
    } catch(e) {
        stackRestore(sp);
        if (e !== e+0) throw e;
        _setThrew(1, 0);
    }
}

function invoke_vijjiiii(index,a1,a2,a3,a4,a5,a6,a7,a8) {
    var sp = stackSave();
    try {
        dynCall_vijjiiii(index,a1,a2,a3,a4,a5,a6,a7,a8);
    } catch(e) {
        stackRestore(sp);
        if (e !== e+0) throw e;
        _setThrew(1, 0);
    }
}

function invoke_ijjjjii(index,a1,a2,a3,a4,a5,a6,a7,a8) {
    var sp = stackSave();
    try {
        return dynCall_ijjjjii(index,a1,a2,a3,a4,a5,a6,a7,a8);
    } catch(e) {
        stackRestore(sp);
        if (e !== e+0) throw e;
        _setThrew(1, 0);
    }
}

function invoke_ijjji(index,a1,a2,a3,a4,a5,a6,a7) {
    var sp = stackSave();
    try {
        return dynCall_ijjji(index,a1,a2,a3,a4,a5,a6,a7);
    } catch(e) {
        stackRestore(sp);
        if (e !== e+0) throw e;

```

```

        _setThrew(1, 0);
    }
}

function invoke_ijjjfi(index,a1,a2,a3,a4,a5,a6,a7,a8) {
    var sp = stackSave();
    try {
        return dynCall_ijjjfi(index,a1,a2,a3,a4,a5,a6,a7,a8);
    } catch(e) {
        stackRestore(sp);
        if (e !== e+0) throw e;
        _setThrew(1, 0);
    }
}

function invoke_ijjjf(index,a1,a2,a3,a4,a5,a6,a7) {
    var sp = stackSave();
    try {
        return dynCall_ijjjf(index,a1,a2,a3,a4,a5,a6,a7);
    } catch(e) {
        stackRestore(sp);
        if (e !== e+0) throw e;
        _setThrew(1, 0);
    }
}

function invoke_ijjjjii(index,a1,a2,a3,a4,a5,a6,a7,a8,a9,a10) {
    var sp = stackSave();
    try {
        return dynCall_ijjjjii(index,a1,a2,a3,a4,a5,a6,a7,a8,a9,a10);
    } catch(e) {
        stackRestore(sp);
        if (e !== e+0) throw e;
        _setThrew(1, 0);
    }
}

function invoke_ijjjji(index,a1,a2,a3,a4,a5,a6,a7,a8,a9) {
    var sp = stackSave();
    try {
        return dynCall_ijjjji(index,a1,a2,a3,a4,a5,a6,a7,a8,a9);
    } catch(e) {
        stackRestore(sp);
        if (e !== e+0) throw e;
        _setThrew(1, 0);
    }
}

function invoke_ijjjiiii(index,a1,a2,a3,a4,a5,a6,a7,a8,a9,a10,a11) {
    var sp = stackSave();
    try {
        return dynCall_ijjjiiii(index,a1,a2,a3,a4,a5,a6,a7,a8,a9,a10,a11);
    } catch(e) {
        stackRestore(sp);
        if (e !== e+0) throw e;
    }
}

```

```

        _setThrew(1, 0);
    }
}

function invoke_ijjjiiii(index,a1,a2,a3,a4,a5,a6,a7,a8,a9,a10) {
    var sp = stackSave();
    try {
        return dynCall_ijjjiiii(index,a1,a2,a3,a4,a5,a6,a7,a8,a9,a10);
    } catch(e) {
        stackRestore(sp);
        if (e !== e+0) throw e;
        _setThrew(1, 0);
    }
}

```

```

// === Auto-generated postamble setup entry stuff ===

unexportedRuntimeFunction('intArrayFromString', false);
unexportedRuntimeFunction('intArrayToString', false);
Module["ccall"] = ccall;
Module["cwrap"] = cwrap;
unexportedRuntimeFunction('setValue', false);
unexportedRuntimeFunction('getValue', false);
unexportedRuntimeFunction('allocate', false);
unexportedRuntimeFunction('UTF8ArrayToString', false);
unexportedRuntimeFunction('UTF8ToString', false);
unexportedRuntimeFunction('stringToUTF8Array', false);
unexportedRuntimeFunction('stringToUTF8', false);
unexportedRuntimeFunction('lengthBytesUTF8', false);
Module["stackTrace"] = stackTrace;
unexportedRuntimeFunction('addOnPreRun', false);
unexportedRuntimeFunction('addOnInit', false);
unexportedRuntimeFunction('addOnPreMain', false);
unexportedRuntimeFunction('addOnExit', false);
unexportedRuntimeFunction('addOnPostRun', false);
unexportedRuntimeFunction('writeStringToMemory', false);
unexportedRuntimeFunction('writeArrayToMemory', false);
unexportedRuntimeFunction('writeAsciiToMemory', false);
Module["addRunDependency"] = addRunDependency;
Module["removeRunDependency"] = removeRunDependency;
unexportedRuntimeFunction('FS_createFolder', false);
Module["FS_createPath"] = FS.createPath;
Module["FS_createDataFile"] = FS.createDataFile;
unexportedRuntimeFunction('FS_createPreloadedFile', true);
unexportedRuntimeFunction('FS_createLazyFile', true);
unexportedRuntimeFunction('FS_createLink', false);
unexportedRuntimeFunction('FS_createDevice', true);
unexportedRuntimeFunction('FS_unlink', true);
unexportedRuntimeFunction('getLEB', false);
unexportedRuntimeFunction('getFunctionTables', false);
unexportedRuntimeFunction('alignFunctionTables', false);
unexportedRuntimeFunction('registerFunctions', false);

```

```
unexportedRuntimeFunction('addFunction', false);
unexportedRuntimeFunction('removeFunction', false);
unexportedRuntimeFunction('getFuncWrapper', false);
unexportedRuntimeFunction('prettyPrint', false);
unexportedRuntimeFunction('dynCall', false);
unexportedRuntimeFunction('getCompilerSetting', false);
unexportedRuntimeFunction('print', false);
unexportedRuntimeFunction('printErr', false);
unexportedRuntimeFunction('getTempRet0', false);
unexportedRuntimeFunction('setTempRet0', false);
unexportedRuntimeFunction('callMain', false);
unexportedRuntimeFunction('abort', false);
unexportedRuntimeFunction('keepRuntimeAlive', false);
unexportedRuntimeFunction('zeroMemory', false);
unexportedRuntimeFunction('stringToNewUTF8', false);
unexportedRuntimeFunction('emscripten_realloc_buffer', false);
unexportedRuntimeFunction('ENV', false);
unexportedRuntimeFunction('ERRNO_CODES', false);
unexportedRuntimeFunction('ERRNO_MESSAGES', false);
unexportedRuntimeFunction('setErrNo', false);
unexportedRuntimeFunction('inetPton4', false);
unexportedRuntimeFunction('inetNtop4', false);
unexportedRuntimeFunction('inetPton6', false);
unexportedRuntimeFunction('inetNtop6', false);
unexportedRuntimeFunction('readSockaddr', false);
unexportedRuntimeFunction('writeSockaddr', false);
unexportedRuntimeFunction('DNS', false);
unexportedRuntimeFunction('getHostByName', false);
unexportedRuntimeFunction('Protocols', false);
unexportedRuntimeFunction('Sockets', false);
unexportedRuntimeFunction('getRandomDevice', false);
unexportedRuntimeFunction('traverseStack', false);
unexportedRuntimeFunction('UNWIND_CACHE', false);
unexportedRuntimeFunction('convertPCtoSourceLocation', false);
unexportedRuntimeFunction('readAsmConstArgsArray', false);
unexportedRuntimeFunction('readAsmConstArgs', false);
unexportedRuntimeFunction('mainThreadEM_ASM', false);
unexportedRuntimeFunction('jstoi_q', false);
unexportedRuntimeFunction('jstoi_s', false);
unexportedRuntimeFunction('getExecutableName', false);
unexportedRuntimeFunction('listenOnce', false);
unexportedRuntimeFunction('autoResumeAudioContext', false);
unexportedRuntimeFunction('dynCallLegacy', false);
unexportedRuntimeFunction('getDynCaller', false);
unexportedRuntimeFunction('dynCall', false);
unexportedRuntimeFunction('handleException', false);
unexportedRuntimeFunction('runtimeKeepalivePush', false);
unexportedRuntimeFunction('runtimeKeepalivePop', false);
unexportedRuntimeFunction('callUserCallback', false);
unexportedRuntimeFunction('maybeExit', false);
unexportedRuntimeFunction('safeSetTimeout', false);
unexportedRuntimeFunction('asmjsMangle', false);
unexportedRuntimeFunction('asyncLoad', false);
unexportedRuntimeFunction('alignMemory', false);
unexportedRuntimeFunction('mmapAlloc', false);
```

```
unexportedRuntimeFunction('reallyNegative', false);
unexportedRuntimeFunction('unSign', false);
unexportedRuntimeFunction('reSign', false);
unexportedRuntimeFunction('formatString', false);
unexportedRuntimeFunction('PATH', false);
unexportedRuntimeFunction('PATH_FS', false);
unexportedRuntimeFunction('SYSCALLS', false);
unexportedRuntimeFunction('getSocketFromFD', false);
unexportedRuntimeFunction('getSocketAddress', false);
unexportedRuntimeFunction('JSEvents', false);
unexportedRuntimeFunction('registerKeyEventCallback', false);
unexportedRuntimeFunction('specialHTMLTargets', false);
unexportedRuntimeFunction('maybeCStringToJsString', false);
unexportedRuntimeFunction('findEventTarget', false);
unexportedRuntimeFunction('findCanvasEventTarget', false);
unexportedRuntimeFunction('getBoundingClientRect', false);
unexportedRuntimeFunction('fillMouseEventData', false);
unexportedRuntimeFunction('registerMouseEventCallback', false);
unexportedRuntimeFunction('registerWheelEventCallback', false);
unexportedRuntimeFunction('registerUiEventCallback', false);
unexportedRuntimeFunction('registerFocusEventCallback', false);
unexportedRuntimeFunction('fillDeviceOrientationEventData', false);
unexportedRuntimeFunction('registerDeviceOrientationEventCallback', false);
unexportedRuntimeFunction('fillDeviceMotionEventData', false);
unexportedRuntimeFunction('registerDeviceMotionEventCallback', false);
unexportedRuntimeFunction('screenOrientation', false);
unexportedRuntimeFunction('fillOrientationChangeEventData', false);
unexportedRuntimeFunction('registerOrientationChangeEventCallback', false);
unexportedRuntimeFunction('fillFullscreenChangeEventData', false);
unexportedRuntimeFunction('registerFullscreenChangeEventCallback', false);
unexportedRuntimeFunction('registerRestoreOldStyle', false);
unexportedRuntimeFunction('hideEverythingExceptGivenElement', false);
unexportedRuntimeFunction('restoreHiddenElements', false);
unexportedRuntimeFunction('setLetterbox', false);
unexportedRuntimeFunction('currentFullscreenStrategy', false);
unexportedRuntimeFunction('restoreOldWindowedStyle', false);
unexportedRuntimeFunction('softFullscreenResizeWebGLRenderTarget', false);
unexportedRuntimeFunction('doRequestFullscreen', false);
unexportedRuntimeFunction('fillPointerlockChangeEventData', false);
unexportedRuntimeFunction('registerPointerlockChangeEventCallback', false);
unexportedRuntimeFunction('registerPointerlockErrorEventCallback', false);
unexportedRuntimeFunction('requestPointerLock', false);
unexportedRuntimeFunction('fillVisibilityChangeEventData', false);
unexportedRuntimeFunction('registerVisibilityChangeEventCallback', false);
unexportedRuntimeFunction('registerTouchEventCallback', false);
unexportedRuntimeFunction('fillGamepadEventData', false);
unexportedRuntimeFunction('registerGamepadEventCallback', false);
unexportedRuntimeFunction('registerBeforeUnloadEventCallback', false);
unexportedRuntimeFunction('fillBatteryEventData', false);
unexportedRuntimeFunction('battery', false);
unexportedRuntimeFunction('registerBatteryEventCallback', false);
unexportedRuntimeFunction('setCanvasElementSize', false);
unexportedRuntimeFunction('getCanvasElementSize', false);
unexportedRuntimeFunction('demangle', false);
unexportedRuntimeFunction('demangleAll', false);
```

```
unexportedRuntimeFunction('jsStackTrace', false);
Module["stackTrace"] = stackTrace;
unexportedRuntimeFunction('getEnvStrings', false);
unexportedRuntimeFunction('checkWasiClock', false);
unexportedRuntimeFunction('writeI53ToI64', false);
unexportedRuntimeFunction('writeI53ToI64Clamped', false);
unexportedRuntimeFunction('writeI53ToI64Signaling', false);
unexportedRuntimeFunction('writeI53ToU64Clamped', false);
unexportedRuntimeFunction('writeI53ToU64Signaling', false);
unexportedRuntimeFunction('readI53FromI64', false);
unexportedRuntimeFunction('readI53FromU64', false);
unexportedRuntimeFunction('convertI32PairToI53', false);
unexportedRuntimeFunction('convertU32PairToI53', false);
unexportedRuntimeFunction('setImmediateWrapped', false);
unexportedRuntimeFunction('clearImmediateWrapped', false);
unexportedRuntimeFunction('polyfillSetImmediate', false);
unexportedRuntimeFunction('uncaughtExceptionCount', false);
unexportedRuntimeFunction('exceptionLast', false);
unexportedRuntimeFunction('exceptionCaught', false);
unexportedRuntimeFunction('ExceptionInfo', false);
unexportedRuntimeFunction('CatchInfo', false);
unexportedRuntimeFunction('exception_addRef', false);
unexportedRuntimeFunction('exception_decRef', false);
unexportedRuntimeFunction('formatException', false);
unexportedRuntimeFunction('Browser', false);
unexportedRuntimeFunction('funcWrappers', false);
unexportedRuntimeFunction('getFuncWrapper', false);
unexportedRuntimeFunction('setMainLoop', false);
unexportedRuntimeFunction('wget', false);
unexportedRuntimeFunction('FS', false);
unexportedRuntimeFunction('MEMFS', false);
unexportedRuntimeFunction('TTY', false);
unexportedRuntimeFunction('PIPEFS', false);
unexportedRuntimeFunction('SOCKFS', false);
unexportedRuntimeFunction('_setNetworkCallback', false);
unexportedRuntimeFunction('tempFixedLengthArray', false);
unexportedRuntimeFunction('miniTempWebGLFloatBuffers', false);
unexportedRuntimeFunction('heapObjectForWebGLType', false);
unexportedRuntimeFunction('heapAccessShiftForWebGLHeap', false);
unexportedRuntimeFunction('GL', false);
unexportedRuntimeFunction('emscriptenWebGLGet', false);
unexportedRuntimeFunction('computeUnpackAlignedImageSize', false);
unexportedRuntimeFunction('emscriptenWebGLGetTexPixelData', false);
unexportedRuntimeFunction('emscriptenWebGLGetUniform', false);
unexportedRuntimeFunction('webglGetUniformLocation', false);
unexportedRuntimeFunction('webglPrepareUniformLocationsBeforeFirstUse', false);
unexportedRuntimeFunction('webglGetLeftBracePos', false);
unexportedRuntimeFunction('emscriptenWebGLGetVertexAttrib', false);
unexportedRuntimeFunction('webglApplyExplicitProgramBindings', false);
unexportedRuntimeFunction('emscriptenWebGLGetBufferBinding', false);
unexportedRuntimeFunction('emscriptenWebGLValidateMapBufferTarget', false);
unexportedRuntimeFunction('writeGLArray', false);
unexportedRuntimeFunction('AL', false);
unexportedRuntimeFunction('SDL_unicode', false);
unexportedRuntimeFunction('SDL_ttfContext', false);
```

```
unexportedRuntimeFunction('SDL_audio', false);
unexportedRuntimeFunction('SDL', false);
unexportedRuntimeFunction('SDL_gfx', false);
unexportedRuntimeFunction('GLUT', false);
unexportedRuntimeFunction('EGL', false);
unexportedRuntimeFunction('GLFW_Window', false);
unexportedRuntimeFunction('GLFW', false);
unexportedRuntimeFunction('GLEW', false);
unexportedRuntimeFunction('IDBStore', false);
unexportedRuntimeFunction('runAndAbortIfError', false);
unexportedRuntimeFunction('emsdkGetIndexed', false);
unexportedRuntimeFunction('remove_cpp_comments_in_shaders', false);
unexportedRuntimeFunction('find_closing_parens_index', false);
unexportedRuntimeFunction('preprocess_c_code', false);
unexportedRuntimeFunction('WEBAudio', false);
unexportedRuntimeFunction('WEBAudio_user', false);
unexportedRuntimeFunction('jsAudioAddPendingBlockedAudio', false);
unexportedRuntimeFunction('jsAudioAddPendingBlockedAudio_user', false);
unexportedRuntimeFunction('jsAudioPlayPendingBlockedAudio', false);
unexportedRuntimeFunction('jsAudioPlayPendingBlockedAudio_user', false);
unexportedRuntimeFunction('jsAudioPlayBlockedAudios', false);
unexportedRuntimeFunction('jsAudioPlayBlockedAudios_user', false);
unexportedRuntimeFunction('jsAudioMixinSetPitch', false);
unexportedRuntimeFunction('jsAudioMixinSetPitch_user', false);
unexportedRuntimeFunction('jsAudioGetMimeTypeFromType', false);
unexportedRuntimeFunction('jsAudioGetMimeTypeFromType_user', false);
unexportedRuntimeFunction('jsAudioCreateCompressedSoundClip', false);
unexportedRuntimeFunction('jsAudioCreateCompressedSoundClip_user', false);
unexportedRuntimeFunction('jsAudioCreateUncompressedSoundClip', false);
unexportedRuntimeFunction('jsAudioCreateUncompressedSoundClip_user', false);
unexportedRuntimeFunction('jsAudioCreateUncompressedSoundClipFromPCM', false);
unexportedRuntimeFunction('jsAudioCreateUncompressedSoundClipFromPCM_user', false);
unexportedRuntimeFunction('jsAudioCreateUncompressedSoundClipFromCompressedAudio',
false);
unexportedRuntimeFunction('jsAudioCreateUncompressedSoundClipFromCompressedAudio_user',
false);
unexportedRuntimeFunction('jsAudioCreateChannel', false);
unexportedRuntimeFunction('jsAudioCreateChannel_user', false);
unexportedRuntimeFunction('registerTouchEventCallback_user', false);
unexportedRuntimeFunction('dlopen_main_init', false);
unexportedRuntimeFunction('dlopen_main_init_user', false);
unexportedRuntimeFunction('jsDomCssEscapeId', false);
unexportedRuntimeFunction('jsDomCssEscapeId_user', false);
unexportedRuntimeFunction('jsCanvasSelector', false);
unexportedRuntimeFunction('jsCanvasSelector_user', false);
unexportedRuntimeFunction('fs', false);
unexportedRuntimeFunction('fs_user', false);
unexportedRuntimeFunction('mobile_input', false);
unexportedRuntimeFunction('mobile_input_user', false);
unexportedRuntimeFunction('mobile_input_text', false);
unexportedRuntimeFunction('mobile_input_text_user', false);
unexportedRuntimeFunction('mobile_input_hide_delay', false);
unexportedRuntimeFunction('mobile_input_hide_delay_user', false);
unexportedRuntimeFunction('mobile_input_ignore_blur_event', false);
unexportedRuntimeFunction('mobile_input_ignore_blur_event_user', false);
```

```
unexportedRuntimeFunction('IDBFS', false);
unexportedRuntimeFunction('JS_ScreenOrientation_callback', false);
unexportedRuntimeFunction('JS_ScreenOrientation_callback_user', false);
unexportedRuntimeFunction('JS_ScreenOrientation_eventHandler', false);
unexportedRuntimeFunction('JS_ScreenOrientation_eventHandler_user', false);
unexportedRuntimeFunction('JS_ScreenOrientation_requestedLockType', false);
unexportedRuntimeFunction('JS_ScreenOrientation_requestedLockType_user', false);
unexportedRuntimeFunction('JS_ScreenOrientation_appliedLockType', false);
unexportedRuntimeFunction('JS_ScreenOrientation_appliedLockType_user', false);
unexportedRuntimeFunction('JS_ScreenOrientation_timeoutID', false);
unexportedRuntimeFunction('JS_ScreenOrientation_timeoutID_user', false);
unexportedRuntimeFunction('JS_OrientationSensor_frequencyRequest', false);
unexportedRuntimeFunction('JS_OrientationSensor_frequencyRequest_user', false);
unexportedRuntimeFunction('JS_OrientationSensor_callback', false);
unexportedRuntimeFunction('JS_OrientationSensor_callback_user', false);
unexportedRuntimeFunction('JS_OrientationSensor', false);
unexportedRuntimeFunction('JS_OrientationSensor_user', false);
unexportedRuntimeFunction('JS_Accelerometer_frequencyRequest', false);
unexportedRuntimeFunction('JS_Accelerometer_frequencyRequest_user', false);
unexportedRuntimeFunction('JS_Accelerometer_callback', false);
unexportedRuntimeFunction('JS_Accelerometer_callback_user', false);
unexportedRuntimeFunction('JS_Accelerometer', false);
unexportedRuntimeFunction('JS_Accelerometer_user', false);
unexportedRuntimeFunction('JS_Accelerometer_multiplier', false);
unexportedRuntimeFunction('JS_Accelerometer_multiplier_user', false);
unexportedRuntimeFunction('JS_LinearAccelerationSensor_frequencyRequest', false);
unexportedRuntimeFunction('JS_LinearAccelerationSensor_frequencyRequest_user', false);
unexportedRuntimeFunction('JS_LinearAccelerationSensor_callback', false);
unexportedRuntimeFunction('JS_LinearAccelerationSensor_callback_user', false);
unexportedRuntimeFunction('JS_LinearAccelerationSensor', false);
unexportedRuntimeFunction('JS_LinearAccelerationSensor_user', false);
unexportedRuntimeFunction('JS_GravitySensor_frequencyRequest', false);
unexportedRuntimeFunction('JS_GravitySensor_frequencyRequest_user', false);
unexportedRuntimeFunction('JS_GravitySensor_callback', false);
unexportedRuntimeFunction('JS_GravitySensor_callback_user', false);
unexportedRuntimeFunction('JS_GravitySensor', false);
unexportedRuntimeFunction('JS_GravitySensor_user', false);
unexportedRuntimeFunction('JS_Accelerometer_frequency', false);
unexportedRuntimeFunction('JS_Accelerometer_frequency_user', false);
unexportedRuntimeFunction('JS_Accelerometer_lastValue', false);
unexportedRuntimeFunction('JS_Accelerometer_lastValue_user', false);
unexportedRuntimeFunction('JS_LinearAccelerationSensor_frequency', false);
unexportedRuntimeFunction('JS_LinearAccelerationSensor_frequency_user', false);
unexportedRuntimeFunction('JS_Gyroscope_frequencyRequest', false);
unexportedRuntimeFunction('JS_Gyroscope_frequencyRequest_user', false);
unexportedRuntimeFunction('JS_Gyroscope_callback', false);
unexportedRuntimeFunction('JS_Gyroscope_callback_user', false);
unexportedRuntimeFunction('JS_Gyroscope', false);
unexportedRuntimeFunction('JS_Gyroscope_user', false);
unexportedRuntimeFunction('JS_DeviceSensorPermissions', false);
unexportedRuntimeFunction('JS_DeviceSensorPermissions_user', false);
unexportedRuntimeFunction('JS_DefineAccelerometerMultiplier', false);
unexportedRuntimeFunction('JS_DefineAccelerometerMultiplier_user', false);
unexportedRuntimeFunction('JS_RequestDeviceSensorPermissions', false);
unexportedRuntimeFunction('JS_RequestDeviceSensorPermissions_user', false);
```

```
unexportedRuntimeFunction('JS_OrientationSensor_eventHandler', false);
unexportedRuntimeFunction('JS_OrientationSensor_eventHandler_user', false);
unexportedRuntimeFunction('JS_Accelerometer_eventHandler', false);
unexportedRuntimeFunction('JS_Accelerometer_eventHandler_user', false);
unexportedRuntimeFunction('JS_ComputeGravity', false);
unexportedRuntimeFunction('JS_ComputeGravity_user', false);
unexportedRuntimeFunction('JS_LinearAccelerationSensor_eventHandler', false);
unexportedRuntimeFunction('JS_LinearAccelerationSensor_eventHandler_user', false);
unexportedRuntimeFunction('JS_GravitySensor_eventHandler', false);
unexportedRuntimeFunction('JS_GravitySensor_eventHandler_user', false);
unexportedRuntimeFunction('JS_Gyroscope_eventHandler', false);
unexportedRuntimeFunction('JS_Gyroscope_eventHandler_user', false);
unexportedRuntimeFunction('JS_DeviceOrientation_eventHandler', false);
unexportedRuntimeFunction('JS_DeviceOrientation_eventHandler_user', false);
unexportedRuntimeFunction('JS_DeviceMotion_eventHandler', false);
unexportedRuntimeFunction('JS_DeviceMotion_eventHandler_user', false);
unexportedRuntimeFunction('JS_DeviceMotion_add', false);
unexportedRuntimeFunction('JS_DeviceMotion_add_user', false);
unexportedRuntimeFunction('JS_DeviceMotion_remove', false);
unexportedRuntimeFunction('JS_DeviceMotion_remove_user', false);
unexportedRuntimeFunction('UNETWebSocketsInstances', false);
unexportedRuntimeFunction('UNETWebSocketsInstances_user', false);
unexportedRuntimeFunction('videoInstances', false);
unexportedRuntimeFunction('videoInstances_user', false);
unexportedRuntimeFunction('videoInstanceIdCounter', false);
unexportedRuntimeFunction('videoInstanceIdCounter_user', false);
unexportedRuntimeFunction('hasSRGBATextures', false);
unexportedRuntimeFunction('hasSRGBATextures_user', false);
unexportedRuntimeFunction('s2lTexture', false);
unexportedRuntimeFunction('s2lTexture_user', false);
unexportedRuntimeFunction('s2lFBO', false);
unexportedRuntimeFunction('s2lFBO_user', false);
unexportedRuntimeFunction('s2lVBO', false);
unexportedRuntimeFunction('s2lVBO_user', false);
unexportedRuntimeFunction('s2lProgram', false);
unexportedRuntimeFunction('s2lProgram_user', false);
unexportedRuntimeFunction('s2lVertexPositionNDC', false);
unexportedRuntimeFunction('s2lVertexPositionNDC_user', false);
unexportedRuntimeFunction('jsVideoEnded', false);
unexportedRuntimeFunction('jsVideoEnded_user', false);
unexportedRuntimeFunction('jsVideoAllAudioTracksAreDisabled', false);
unexportedRuntimeFunction('jsVideoAllAudioTracksAreDisabled_user', false);
unexportedRuntimeFunction('jsVideoPendingBlockedVideos', false);
unexportedRuntimeFunction('jsVideoPendingBlockedVideos_user', false);
unexportedRuntimeFunction('jsVideoAddPendingBlockedVideo', false);
unexportedRuntimeFunction('jsVideoAddPendingBlockedVideo_user', false);
unexportedRuntimeFunction('jsVideoPlayPendingBlockedVideo', false);
unexportedRuntimeFunction('jsVideoPlayPendingBlockedVideo_user', false);
unexportedRuntimeFunction('jsVideoRemovePendingBlockedVideo', false);
unexportedRuntimeFunction('jsVideoRemovePendingBlockedVideo_user', false);
unexportedRuntimeFunction('jsVideoAttemptToPlayBlockedVideos', false);
unexportedRuntimeFunction('jsVideoAttemptToPlayBlockedVideos_user', false);
unexportedRuntimeFunction('jsVideoCreateTexture2D', false);
unexportedRuntimeFunction('jsVideoCreateTexture2D_user', false);
unexportedRuntimeFunction('jsSupportedVideoFormats', false);
```

```

unexportedRuntimeFunction('jsSupportedVideoFormats__user', false);
unexportedRuntimeFunction('jsUnsupportedVideoFormats', false);
unexportedRuntimeFunction('jsUnsupportedVideoFormats__user', false);
unexportedRuntimeFunction('activeWebCams', false);
unexportedRuntimeFunction('activeWebCams__user', false);
unexportedRuntimeFunction('cameraAccess', false);
unexportedRuntimeFunction('cameraAccess__user', false);
unexportedRuntimeFunction('wr', false);
unexportedRuntimeFunction('wr__user', false);
unexportedRuntimeFunction('jsWebRequestGetResponseHeaderString', false);
unexportedRuntimeFunction('jsWebRequestGetResponseHeaderString__user', false);
unexportedRuntimeFunction('warnOnce', false);
unexportedRuntimeFunction('stackSave', false);
unexportedRuntimeFunction('stackRestore', false);
unexportedRuntimeFunction('stackAlloc', false);
unexportedRuntimeFunction('AsciiToString', false);
unexportedRuntimeFunction('stringToAscii', false);
unexportedRuntimeFunction('UTF16ToString', false);
unexportedRuntimeFunction('stringToUTF16', false);
unexportedRuntimeFunction('lengthBytesUTF16', false);
unexportedRuntimeFunction('UTF32ToString', false);
unexportedRuntimeFunction('stringToUTF32', false);
unexportedRuntimeFunction('lengthBytesUTF32', false);
unexportedRuntimeFunction('allocateUTF8', false);
unexportedRuntimeFunction('allocateUTF8OnStack', false);
Module["writeStackCookie"] = writeStackCookie;
Module["checkStackCookie"] = checkStackCookie;
unexportedRuntimeSymbol('ALLOC_NORMAL', false);
unexportedRuntimeSymbol('ALLOC_STACK', false);

var calledRun;

/**
 * @constructor
 * @this {ExitStatus}
 */
function ExitStatus(status) {
  this.name = "ExitStatus";
  this.message = "Program terminated with exit(" + status + ")";
  this.status = status;
}

var calledMain = false;

dependenciesFulfilled = function runCaller() {
  // If run has never been called, and we should call run (INVOKE_RUN is true, and
  // Module.noInitialRun is not false)
  if (!calledRun) run();
  if (!calledRun) dependenciesFulfilled = runCaller; // try this again later, after new
  deps are fulfilled
};

function callMain(args) {
  assert(runDependencies == 0, 'cannot call main when async dependencies remain!
  (listen on Module["onRuntimeInitialized"]))');
}

```

```

assert(__ATPRERUN__.length == 0, 'cannot call main when preRun functions remain to be
called');

var entryFunction = Module['__main'];

args = args || [];

var argc = args.length+1;
var argv = stackAlloc((argc + 1) * 4);
HEAP32[argv >> 2] = allocateUTF8OnStack(thisProgram);
for (var i = 1; i < argc; i++) {
    HEAP32[(argv >> 2) + i] = allocateUTF8OnStack(args[i - 1]);
}
HEAP32[(argv >> 2) + argc] = 0;

try {
    var ret = entryFunction(argc, argv);

    // In PROXY_TO_PTHREAD builds, we should never exit the runtime below, as
    // execution is asynchronously handed off to a pthread.
    // if we're not running an evented main loop, it's time to exit
    exit(ret, /* implicit = */ true);
    return ret;
}
catch (e) {
    return handleException(e);
} finally {
    calledMain = true;
}

function stackCheckInit() {
    // This is normally called automatically during __wasm_call_ctors but need to
    // get these values before even running any of the ctors so we call it redundantly
    // here.
    // TODO(sbc): Move writeStackCookie to native to to avoid this.
    _emscripten_stack_init();
    writeStackCookie();
}

/** @type {function(Array=)} */
function run(args) {
    args = args || arguments_;

    if (runDependencies > 0) {
        return;
    }

    stackCheckInit();

    preRun();

    // a preRun added a dependency, run will be called later
}

```

```

if (runDependencies > 0) {
    return;
}

function doRun() {
    // run may have just been called through dependencies being fulfilled just in this
    // very frame,
    // or while the async setStatus time below was happening
    if (calledRun) return;
    calledRun = true;
    Module['calledRun'] = true;

    if (ABORT) return;

    initRuntime();

    preMain();

    readyPromiseResolve(Module);
    if (Module['onRuntimeInitialized']) Module['onRuntimeInitialized']();

    if (shouldRunNow) callMain(args);

    postRun();
}

if (Module['setStatus']) {
    Module['setStatus']('Running...');
    setTimeout(function() {
        setTimeout(function() {
            Module['setStatus']( '');
        }, 1);
        doRun();
    }, 1);
} else
{
    doRun();
}
checkStackCookie();
}

Module['run'] = run;

function checkUnflushedContent() {
    // Compiler settings do not allow exiting the runtime, so flushing
    // the streams is not possible. but in ASSERTIONS mode we check
    // if there was something to flush, and if so tell the user they
    // should request that the runtime be exitable.
    // Normally we would not even include flush() at all, but in ASSERTIONS
    // builds we do so just for this check, and here we see if there is any
    // content to flush, that is, we check if there would have been
    // something a non-ASSERTIONS build would have not seen.
    // How we flush the streams depends on whether we are in
SYSCALLS_REQUIRE_FILESYSTEM=0
    // mode (which has its own special function for this; otherwise, all
    // the code is inside libc)
}

```

```

var oldOut = out;
var oldErr = err;
var has = false;
out = err = (x) => {
    has = true;
}
try { // it doesn't matter if it fails
    __stdio_exit();
    // also flush in the JS FS layer
    ['stdout', 'stderr'].forEach(function(name) {
        var info = FS.analyzePath('/dev/' + name);
        if (!info) return;
        var stream = info.object;
        var rdev = stream.rdev;
        var tty = TTY.ttys[rdev];
        if (tty && tty.output && tty.output.length) {
            has = true;
        }
    });
} catch(e) {}
out = oldOut;
err = oldErr;
if (has) {
    warnOnce('stdio streams had content in them that was not flushed. you should set
EXIT_RUNTIME to 1 (see the FAQ), or make sure to emit a newline when you printf etc.');
}
}

/** @param {boolean|number=} implicit */
function exit(status, implicit) {
    EXITSTATUS = status;

    checkUnflushedContent();

    // if exit() was called explicitly, warn the user if the runtime isn't actually being
    // shut down
    if (keepRuntimeAlive() && !implicit) {
        var msg = 'program exited (with status: ' + status + '), but EXIT_RUNTIME is not
set, so halting execution but not exiting the runtime or preventing further async
execution (build with EXIT_RUNTIME=1, if you want a true shutdown)';
        readyPromiseReject(msg);
        err(msg);
    }

    procExit(status);
}

function procExit(code) {
    EXITSTATUS = code;
    if (!keepRuntimeAlive()) {
        if (Module['onExit']) Module['onExit'](code);
        ABORT = true;
    }
    quit_(code, new ExitStatus(code));
}

```

```
if (Module['preInit']) {
  if (typeof Module['preInit'] == 'function') Module['preInit'] = [Module['preInit']];
  while (Module['preInit'].length > 0) {
    Module['preInit'].pop()();
  }
}

// shouldRunNow refers to calling main(), not run().
var shouldRunNow = true;

if (Module['noInitialRun']) shouldRunNow = false;

run();

return unityFramework.ready
};

})();
if (typeof exports === 'object' && typeof module === 'object')
  module.exports = unityFramework;
else if (typeof define === 'function' && define['amd'])
  define([], function() { return unityFramework; });
else if (typeof exports === 'object')
  exports["unityFramework"] = unityFramework;
```