Python WebAssembly (WASM) build

WARNING: WASM support is highly experimental! Lots of features are not working yet.

This directory contains configuration and helpers to facilitate cross compilation of CPython to WebAssembly (WASM). For now we support *wasm32-emscripten* builds for modern browser and for *Node.js*. It's not possible to build for *wasm32-wasi* out-of-the-box yet.

wasm32-emscripten build

Cross compiling to wasm32-emscripten platform needs the Emscripten tool chain and a build Python interpreter. All commands below are relative to a repository checkout.

Compile a build Python interpreter

```
mkdir -p builddir/build
pushd builddir/build
../../configure -C
make -j$(nproc)
popd
```

Fetch and build additional emscripten ports

```
embuilder build zlib bzip2
```

Cross compile to wasm32-emscripten for browser

```
mkdir -p builddir/emscripten-browser
pushd builddir/emscripten-browser

CONFIG_SITE=../../Tools/wasm/config.site-wasm32-emscripten \
  emconfigure ../../configure -C \
    --host=wasm32-unknown-emscripten \
    --build=$(../../config.guess) \
    --with-emscripten-target=browser \
    --with-build-python=$(pwd)/../build/python

emmake make -j$(nproc)
popd
```

Serve `python.html` with a local webserver and open the file in a browser.

```
emrun builddir/emscripten-browser/python.html
```

or

```
./Tools/wasm/wasm_webserver.py
```

and open http://localhost:8000/builddir/emscripten-browser/python.html . This directory structure enables the *C/C++ DevTools Support (DWARF)* to load C and header files with debug builds.

Cross compile to wasm32-emscripten for node

```
mkdir -p builddir/emscripten-node
pushd builddir/emscripten-node

CONFIG_SITE=../../Tools/wasm/config.site-wasm32-emscripten \
  emconfigure ../../configure -C \
    --host=wasm32-unknown-emscripten \
    --build=$(../../config.guess) \
    --with-emscripten-target=node \
    --with-build-python=$(pwd)/../build/python

emmake make -j$(nproc)
popd
```

```
node --experimental-wasm-threads --experimental-wasm-bulk-memory builddir/emscripten-node/python.js
```

wasm32-emscripten limitations and issues

Emscripten before 3.1.8 has known bugs that can cause memory corruption and resource leaks. 3.1.8 contains several fixes for bugs in date and time functions.

Network stack

- Python's socket module does not work with Emscripten's emulated POSIX sockets yet. Network modules like `asyncio` , `urllib` , `selectors` , etc. are not available.
- Only `AF_INET` and `AF_INET6` with `SOCK_STREAM` (TCP) or `SOCK_DGRAM` (UDP) are available. `AF_UNIX` is not supported.
- `socketpair` does not work.
- Blocking sockets are not available and non-blocking sockets don't work correctly, e.g. `socket.accept` crashes the runtime. `gethostbyname` does not resolve to a real IP address. IPv6 is not available.
- The `select` module is limited. `select.select()` crashes the runtime due to lack of exectfd support.

processes, threads, signals

- Processes are not supported. System calls like fork, popen, and subprocess fail with `ENOSYS` or `ENOSUP` .
- Signal support is limited. `signal.alarm` , `itimer` , `sigaction` are not available or do not work correctly. `SIGTERM` exits the runtime.
- Keyboard interrupt (CTRL+C) handling is not implemented yet.
- Browser builds cannot start new threads. Node's web workers consume extra file descriptors.
- Resource-related functions like `os.nice` and most functions of the `resource` module are not available.

file system

- Most user, group, and permission related function and modules are not supported or don't work as expected, e.g. `pwd` module, `grp` module, `os.setgroups`, `os.chown`, and so on. `lchown` and `lchmod`` are not available.
- `umask` is a no-op.
- hard links (`os.link`) are not supported.
- Offset and iovec I/O functions (e.g. `os.pread`, `os.preadv`) are not available.
- `os.mknod` and `os.mkfifo` don't work and are disabled.
- Large file support crashes the runtime and is disabled.
- `mmap` module is unstable. flush (`msync`) can crash the runtime.

Misc

- Heap memory and stack size are limited. Recursion or extensive memory consumption can crash Python.
- Most stdlib modules with a dependency on external libraries are missing, e.g. `ctypes`, `readline`, `sqlite3`, `ssl`, and more.
- Shared extension modules are not implemented yet. All extension modules are statically linked into the main binary. The experimental configure option `--enable-wasm-dynamic-linking` enables dynamic extensions.
- glibc extensions for date and time formatting are not available.
- `locales` module is affected by musl libc issues, bpo-46390.
- Python's object allocator `obmalloc` is disabled by default.
- `ensurepip` is not available.

wasm32-emscripten in browsers

- The interactive shell does not handle copy 'n paste and unicode support well.
- The bundled stdlib is limited. Network-related modules, distutils, multiprocessing, dbm, tests and similar modules are not shipped. All other modules are bundled as pre-compiled `pyc` files.
- Threading is not supported.
- In-memory file system (MEMFS) is not persistent and limited.

wasm32-emscripten in node

Node builds use `NODERAWFS`, `USE_PTHREADS` and `PROXY_TO_PTHREAD` linker options.

- Node RawFS allows direct access to the host file system.
- pthread support requires WASM threads and SharedArrayBuffer (bulk memory). The runtime keeps a pool of web workers around. Each web worker uses several file descriptors (eventfd, epoll, pipe).