/\*\*

\* The `os` module provides operating system-related utility methods and

\* properties. It can be accessed using:

\*

\* ```js

\* const os = require('os');

\* ```

\* @see [source](https://github.com/nodejs/node/blob/v17.0.0/lib/os.js)

\*/

declare module 'os' {

interface CpuInfo {

model: string;

speed: number;

times: {

user: number;

nice: number;

sys: number;

idle: number;

irq: number;

};

}

interface NetworkInterfaceBase {

address: string;

netmask: string;

mac: string;

internal: boolean;

cidr: string | null;

}

interface NetworkInterfaceInfoIPv4 extends NetworkInterfaceBase {

family: 'IPv4';

}

interface NetworkInterfaceInfoIPv6 extends NetworkInterfaceBase {

family: 'IPv6';

scopeid: number;

}

interface UserInfo<T> {

username: T;

uid: number;

gid: number;

shell: T;

homedir: T;

}

type NetworkInterfaceInfo = NetworkInterfaceInfoIPv4 | NetworkInterfaceInfoIPv6;

/\*\*

\* Returns the host name of the operating system as a string.

\* @since v0.3.3

\*/

function hostname(): string;

/\*\*

\* Returns an array containing the 1, 5, and 15 minute load averages.

\*

\* The load average is a measure of system activity calculated by the operating

\* system and expressed as a fractional number.

\*

\* The load average is a Unix-specific concept. On Windows, the return value is

\* always `[0, 0, 0]`.

\* @since v0.3.3

\*/

function loadavg(): number[];

/\*\*

\* Returns the system uptime in number of seconds.

\* @since v0.3.3

\*/

function uptime(): number;

/\*\*

\* Returns the amount of free system memory in bytes as an integer.

\* @since v0.3.3

\*/

function freemem(): number;

/\*\*

\* Returns the total amount of system memory in bytes as an integer.

\* @since v0.3.3

\*/

function totalmem(): number;

/\*\*

\* Returns an array of objects containing information about each logical CPU core.

\*

\* The properties included on each object include:

\*

\* ```js

\* [

\* {

\* model: 'Intel(R) Core(TM) i7 CPU 860 @ 2.80GHz',

\* speed: 2926,

\* times: {

\* user: 252020,

\* nice: 0,

\* sys: 30340,

\* idle: 1070356870,

\* irq: 0

\* }

\* },

\* {

\* model: 'Intel(R) Core(TM) i7 CPU 860 @ 2.80GHz',

\* speed: 2926,

\* times: {

\* user: 306960,

\* nice: 0,

\* sys: 26980,

\* idle: 1071569080,

\* irq: 0

\* }

\* },

\* {

\* model: 'Intel(R) Core(TM) i7 CPU 860 @ 2.80GHz',

\* speed: 2926,

\* times: {

\* user: 248450,

\* nice: 0,

\* sys: 21750,

\* idle: 1070919370,

\* irq: 0

\* }

\* },

\* {

\* model: 'Intel(R) Core(TM) i7 CPU 860 @ 2.80GHz',

\* speed: 2926,

\* times: {

\* user: 256880,

\* nice: 0,

\* sys: 19430,

\* idle: 1070905480,

\* irq: 20

\* }

\* },

\* ]

\* ```

\*

\* `nice` values are POSIX-only. On Windows, the `nice` values of all processors

\* are always 0.

\* @since v0.3.3

\*/

function cpus(): CpuInfo[];

/\*\*

\* Returns the operating system name as returned by [`uname(3)`](https://linux.die.net/man/3/uname). For example, it

\* returns `'Linux'` on Linux, `'Darwin'` on macOS, and `'Windows\_NT'` on Windows.

\*

\* See [https://en.wikipedia.org/wiki/Uname#Examples](https://en.wikipedia.org/wiki/Uname#Examples) for additional information

\* about the output of running [`uname(3)`](https://linux.die.net/man/3/uname) on various operating systems.

\* @since v0.3.3

\*/

function type(): string;

/\*\*

\* Returns the operating system as a string.

\*

\* On POSIX systems, the operating system release is determined by calling [`uname(3)`](https://linux.die.net/man/3/uname). On Windows, `GetVersionExW()` is used. See

\* [https://en.wikipedia.org/wiki/Uname#Examples](https://en.wikipedia.org/wiki/Uname#Examples) for more information.

\* @since v0.3.3

\*/

function release(): string;

/\*\*

\* Returns an object containing network interfaces that have been assigned a

\* network address.

\*

\* Each key on the returned object identifies a network interface. The associated

\* value is an array of objects that each describe an assigned network address.

\*

\* The properties available on the assigned network address object include:

\*

\* ```js

\* {

\* lo: [

\* {

\* address: '127.0.0.1',

\* netmask: '255.0.0.0',

\* family: 'IPv4',

\* mac: '00:00:00:00:00:00',

\* internal: true,

\* cidr: '127.0.0.1/8'

\* },

\* {

\* address: '::1',

\* netmask: 'ffff:ffff:ffff:ffff:ffff:ffff:ffff:ffff',

\* family: 'IPv6',

\* mac: '00:00:00:00:00:00',

\* scopeid: 0,

\* internal: true,

\* cidr: '::1/128'

\* }

\* ],

\* eth0: [

\* {

\* address: '192.168.1.108',

\* netmask: '255.255.255.0',

\* family: 'IPv4',

\* mac: '01:02:03:0a:0b:0c',

\* internal: false,

\* cidr: '192.168.1.108/24'

\* },

\* {

\* address: 'fe80::a00:27ff:fe4e:66a1',

\* netmask: 'ffff:ffff:ffff:ffff::',

\* family: 'IPv6',

\* mac: '01:02:03:0a:0b:0c',

\* scopeid: 1,

\* internal: false,

\* cidr: 'fe80::a00:27ff:fe4e:66a1/64'

\* }

\* ]

\* }

\* ```

\* @since v0.6.0

\*/

function networkInterfaces(): NodeJS.Dict<NetworkInterfaceInfo[]>;

/\*\*

\* Returns the string path of the current user's home directory.

\*

\* On POSIX, it uses the `$HOME` environment variable if defined. Otherwise it

\* uses the [effective UID](https://en.wikipedia.org/wiki/User\_identifier#Effective\_user\_ID) to look up the user's home directory.

\*

\* On Windows, it uses the `USERPROFILE` environment variable if defined.

\* Otherwise it uses the path to the profile directory of the current user.

\* @since v2.3.0

\*/

function homedir(): string;

/\*\*

\* Returns information about the currently effective user. On POSIX platforms,

\* this is typically a subset of the password file. The returned object includes

\* the `username`, `uid`, `gid`, `shell`, and `homedir`. On Windows, the `uid` and`gid` fields are `-1`, and `shell` is `null`.

\*

\* The value of `homedir` returned by `os.userInfo()` is provided by the operating

\* system. This differs from the result of `os.homedir()`, which queries

\* environment variables for the home directory before falling back to the

\* operating system response.

\*

\* Throws a `SystemError` if a user has no `username` or `homedir`.

\* @since v6.0.0

\*/

function userInfo(options: { encoding: 'buffer' }): UserInfo<Buffer>;

function userInfo(options?: { encoding: BufferEncoding }): UserInfo<string>;

type SignalConstants = {

[key in NodeJS.Signals]: number;

};

namespace constants {

const UV\_UDP\_REUSEADDR: number;

namespace signals {}

const signals: SignalConstants;

namespace errno {

const E2BIG: number;

const EACCES: number;

const EADDRINUSE: number;

const EADDRNOTAVAIL: number;

const EAFNOSUPPORT: number;

const EAGAIN: number;

const EALREADY: number;

const EBADF: number;

const EBADMSG: number;

const EBUSY: number;

const ECANCELED: number;

const ECHILD: number;

const ECONNABORTED: number;

const ECONNREFUSED: number;

const ECONNRESET: number;

const EDEADLK: number;

const EDESTADDRREQ: number;

const EDOM: number;

const EDQUOT: number;

const EEXIST: number;

const EFAULT: number;

const EFBIG: number;

const EHOSTUNREACH: number;

const EIDRM: number;

const EILSEQ: number;

const EINPROGRESS: number;

const EINTR: number;

const EINVAL: number;

const EIO: number;

const EISCONN: number;

const EISDIR: number;

const ELOOP: number;

const EMFILE: number;

const EMLINK: number;

const EMSGSIZE: number;

const EMULTIHOP: number;

const ENAMETOOLONG: number;

const ENETDOWN: number;

const ENETRESET: number;

const ENETUNREACH: number;

const ENFILE: number;

const ENOBUFS: number;

const ENODATA: number;

const ENODEV: number;

const ENOENT: number;

const ENOEXEC: number;

const ENOLCK: number;

const ENOLINK: number;

const ENOMEM: number;

const ENOMSG: number;

const ENOPROTOOPT: number;

const ENOSPC: number;

const ENOSR: number;

const ENOSTR: number;

const ENOSYS: number;

const ENOTCONN: number;

const ENOTDIR: number;

const ENOTEMPTY: number;

const ENOTSOCK: number;

const ENOTSUP: number;

const ENOTTY: number;

const ENXIO: number;

const EOPNOTSUPP: number;

const EOVERFLOW: number;

const EPERM: number;

const EPIPE: number;

const EPROTO: number;

const EPROTONOSUPPORT: number;

const EPROTOTYPE: number;

const ERANGE: number;

const EROFS: number;

const ESPIPE: number;

const ESRCH: number;

const ESTALE: number;

const ETIME: number;

const ETIMEDOUT: number;

const ETXTBSY: number;

const EWOULDBLOCK: number;

const EXDEV: number;

const WSAEINTR: number;

const WSAEBADF: number;

const WSAEACCES: number;

const WSAEFAULT: number;

const WSAEINVAL: number;

const WSAEMFILE: number;

const WSAEWOULDBLOCK: number;

const WSAEINPROGRESS: number;

const WSAEALREADY: number;

const WSAENOTSOCK: number;

const WSAEDESTADDRREQ: number;

const WSAEMSGSIZE: number;

const WSAEPROTOTYPE: number;

const WSAENOPROTOOPT: number;

const WSAEPROTONOSUPPORT: number;

const WSAESOCKTNOSUPPORT: number;

const WSAEOPNOTSUPP: number;

const WSAEPFNOSUPPORT: number;

const WSAEAFNOSUPPORT: number;

const WSAEADDRINUSE: number;

const WSAEADDRNOTAVAIL: number;

const WSAENETDOWN: number;

const WSAENETUNREACH: number;

const WSAENETRESET: number;

const WSAECONNABORTED: number;

const WSAECONNRESET: number;

const WSAENOBUFS: number;

const WSAEISCONN: number;

const WSAENOTCONN: number;

const WSAESHUTDOWN: number;

const WSAETOOMANYREFS: number;

const WSAETIMEDOUT: number;

const WSAECONNREFUSED: number;

const WSAELOOP: number;

const WSAENAMETOOLONG: number;

const WSAEHOSTDOWN: number;

const WSAEHOSTUNREACH: number;

const WSAENOTEMPTY: number;

const WSAEPROCLIM: number;

const WSAEUSERS: number;

const WSAEDQUOT: number;

const WSAESTALE: number;

const WSAEREMOTE: number;

const WSASYSNOTREADY: number;

const WSAVERNOTSUPPORTED: number;

const WSANOTINITIALISED: number;

const WSAEDISCON: number;

const WSAENOMORE: number;

const WSAECANCELLED: number;

const WSAEINVALIDPROCTABLE: number;

const WSAEINVALIDPROVIDER: number;

const WSAEPROVIDERFAILEDINIT: number;

const WSASYSCALLFAILURE: number;

const WSASERVICE\_NOT\_FOUND: number;

const WSATYPE\_NOT\_FOUND: number;

const WSA\_E\_NO\_MORE: number;

const WSA\_E\_CANCELLED: number;

const WSAEREFUSED: number;

}

namespace priority {

const PRIORITY\_LOW: number;

const PRIORITY\_BELOW\_NORMAL: number;

const PRIORITY\_NORMAL: number;

const PRIORITY\_ABOVE\_NORMAL: number;

const PRIORITY\_HIGH: number;

const PRIORITY\_HIGHEST: number;

}

}

const devNull: string;

const EOL: string;

/\*\*

\* Returns the operating system CPU architecture for which the Node.js binary was

\* compiled. Possible values are `'arm'`, `'arm64'`, `'ia32'`, `'mips'`,`'mipsel'`, `'ppc'`, `'ppc64'`, `'s390'`, `'s390x'`, `'x32'`, and `'x64'`.

\*

\* The return value is equivalent to `process.arch`.

\* @since v0.5.0

\*/

function arch(): string;

/\*\*

\* Returns a string identifying the kernel version.

\*

\* On POSIX systems, the operating system release is determined by calling [`uname(3)`](https://linux.die.net/man/3/uname). On Windows, `RtlGetVersion()` is used, and if it is not

\* available, `GetVersionExW()` will be used. See [https://en.wikipedia.org/wiki/Uname#Examples](https://en.wikipedia.org/wiki/Uname#Examples) for more information.

\* @since v13.11.0, v12.17.0

\*/

function version(): string;

/\*\*

\* Returns a string identifying the operating system platform. The value is set

\* at compile time. Possible values are `'aix'`, `'darwin'`, `'freebsd'`,`'linux'`, `'openbsd'`, `'sunos'`, and `'win32'`.

\*

\* The return value is equivalent to `process.platform`.

\*

\* The value `'android'` may also be returned if Node.js is built on the Android

\* operating system. [Android support is experimental](https://github.com/nodejs/node/blob/HEAD/BUILDING.md#androidandroid-based-devices-eg-firefox-os).

\* @since v0.5.0

\*/

function platform(): NodeJS.Platform;

/\*\*

\* Returns the operating system's default directory for temporary files as a

\* string.

\* @since v0.9.9

\*/

function tmpdir(): string;

/\*\*

\* Returns a string identifying the endianness of the CPU for which the Node.js

\* binary was compiled.

\*

\* Possible values are `'BE'` for big endian and `'LE'` for little endian.

\* @since v0.9.4

\*/

function endianness(): 'BE' | 'LE';

/\*\*

\* Returns the scheduling priority for the process specified by `pid`. If `pid` is

\* not provided or is `0`, the priority of the current process is returned.

\* @since v10.10.0

\* @param [pid=0] The process ID to retrieve scheduling priority for.

\*/

function getPriority(pid?: number): number;

/\*\*

\* Attempts to set the scheduling priority for the process specified by `pid`. If`pid` is not provided or is `0`, the process ID of the current process is used.

\*

\* The `priority` input must be an integer between `-20` (high priority) and `19`(low priority). Due to differences between Unix priority levels and Windows

\* priority classes, `priority` is mapped to one of six priority constants in`os.constants.priority`. When retrieving a process priority level, this range

\* mapping may cause the return value to be slightly different on Windows. To avoid

\* confusion, set `priority` to one of the priority constants.

\*

\* On Windows, setting priority to `PRIORITY\_HIGHEST` requires elevated user

\* privileges. Otherwise the set priority will be silently reduced to`PRIORITY\_HIGH`.

\* @since v10.10.0

\* @param [pid=0] The process ID to set scheduling priority for.

\* @param priority The scheduling priority to assign to the process.

\*/

function setPriority(priority: number): void;

function setPriority(pid: number, priority: number): void;

}

declare module 'node:os' {

export \* from 'os';

}