Constants

`Core.nothing` — Constant

```
nothing
```

The singleton instance of type `Nothing`, used by convention when there is no value to return (as in a C `void` function) or when a variable or field holds no value.

`Base.PROGRAM_FILE` — Constant

```
PROGRAM_FILE
```

A string containing the script name passed to Julia from the command line. Note that the script name remains unchanged from within included files. Alternatively see `@__FILE__`.

`Base.ARGS` — Constant

```
ARGS
```

An array of the command line arguments passed to Julia, as strings.

`Base.C_NULL` — Constant

```
C_NULL
```

The C null pointer constant, sometimes used when calling external code.

`Base.VERSION` — Constant

```
VERSION
```

A `VersionNumber` object describing which version of Julia is in use. For details see Version Number Literals.

`Base.DEPOT_PATH` — Constant

```
DEPOT_PATH
```

A stack of "depot" locations where the package manager, as well as Julia's code loading mechanisms, look for package registries, installed packages, named environments, repo clones, cached compiled package images, and configuration files. By default it includes:

1. `~/.julia` where ~ is the user home as appropriate on the system;
2. an architecture-specific shared system directory, e.g. `/usr/local/share/julia`;
3. an architecture-independent shared system directory, e.g. `/usr/share/julia`.

So `DEPOT_PATH` might be:

```
[joinpath(homedir(), ".julia"), "/usr/local/share/julia", "/usr/share/julia"]
```

The first entry is the "user depot" and should be writable by and owned by the current user. The user depot is where: registries are cloned, new package versions are installed, named environments are created and updated, package repos are cloned, newly compiled package image files are saved, log files are written, development packages are checked out by default, and global configuration data is saved. Later entries in the depot path are treated as read-only and are appropriate for registries, packages, etc. installed and managed by system administrators.

`DEPOT_PATH` is populated based on the `JULIA_DEPOT_PATH` environment variable if set.

See also: `JULIA_DEPOT_PATH`, and Code Loading.

`Base.LOAD_PATH` — Constant

```
LOAD_PATH
```

An array of paths for `using` and `import` statements to consider as project environments or package directories when loading code. It is populated based on the `JULIA_LOAD_PATH` environment variable if set; otherwise it defaults to `["@", "@v#.#", "@stdlib"]`. Entries starting with `@` have special meanings:

- `@` refers to the "current active environment", the initial value of which is initially determined by the `JULIA_PROJECT` environment variable or the `--project` command-line option.
- `@stdlib` expands to the absolute path of the current Julia installation's standard library directory.
- `@name` refers to a named environment, which are stored in depots (see `JULIA_DEPOT_PATH`) under the `environments` subdirectory. The user's named environments are stored in `~/.julia/environments` so `@name` would refer to the environment in `~/.julia/environments/name` if it exists and contains a `Project.toml` file. If `name` contains `#` characters, then they are replaced with the major, minor and patch components of the Julia version number. For example, if you are running Julia 1.2 then `@v#.#` expands to `@v1.2` and will look for an environment by that name, typically at `~/.julia/environments/v1.2`.

The fully expanded value of `LOAD_PATH` that is searched for projects and packages can be seen by calling the `Base.load_path()` function.

See also: `JULIA_LOAD_PATH`, `JULIA_PROJECT`, `JULIA_DEPOT_PATH`, and Code Loading.

`Base.Sys.BINDIR` — Constant

```
Sys.BINDIR
```

A string containing the full path to the directory containing the `julia` executable.

`Base.Sys.CPU_THREADS` — Constant

```
Sys.CPU_THREADS
```

The number of logical CPU cores available in the system, i.e. the number of threads that the CPU can run concurrently. Note that this is not necessarily the number of CPU cores, for example, in the presence of hyper-threading.

See Hwloc.jl or CpuId.jl for extended information, including number of physical cores.

Base.Sys.WORD_SIZE — Constant

```
Sys.WORD_SIZE
```

Standard word size on the current machine, in bits.

Base.Sys.KERNEL — Constant

```
Sys.KERNEL
```

A symbol representing the name of the operating system, as returned by `uname` of the build configuration.

Base.Sys.ARCH — Constant

```
Sys.ARCH
```

A symbol representing the architecture of the build configuration.

Base.Sys.MACHINE — Constant

```
Sys.MACHINE
```

A string containing the build triple.

See also:

- stdin
- stdout
- stderr
- ENV
- ENDIAN_BOM
- Libc.MS_ASYNC

- `Libc.MS_INVALIDATE`
- `Libc.MS_SYNC`

« Multi-Threading Filesystem »