Data Type: **DIR**

The `DIR` data type represents a directory stream.

You shouldn't ever allocate objects of the `struct dirent` or `DIR` data types, since the directory access functions do that for you. Instead, you refer to these objects using the pointers returned by the following functions.

Directory streams are a high-level interface. On Linux, alternative interfaces for accessing directories using file descriptors are available. See Low-level Directory Access.

Function: `DIR *` **`opendir`** `(const char *dirname)`

Preliminary: | MT-Safe | AS-Unsafe heap | AC-Unsafe mem fd | See POSIX Safety Concepts.

The `opendir` function opens and returns a directory stream for reading the directory whose file name is *dirname*. The stream has type `DIR *`.

If unsuccessful, `opendir` returns a null pointer. In addition to the usual file name errors (see File Name Errors), the following `errno` error conditions are defined for this function:

`EACCES`

Read permission is denied for the directory named by `dirname`.

`EMFILE`

The process has too many files open.

14.3.3 Reading and Closing a Directory Stream

This section describes how to read directory entries from a directory stream, and how to close the stream when you are done with it. All the symbols are declared in the header file `dirent.h`.

Function: `struct dirent *` **`readdir`** `(DIR *`*`dirstream`*`)`

> Preliminary: | MT-Safe | AS-Unsafe lock | AC-Unsafe lock | See POSIX Safety Concepts.
>
> This function reads the next entry from the directory. It normally returns a pointer to a structure containing information about the file. This structure is associated with the *dirstream* handle and can be rewritten by a subsequent call.
>
> **Portability Note:** On some systems `readdir` may not return entries for `.` and `..`, even though these are always valid file names in any directory. See File Name Resolution.
>
> If there are no more entries in the directory or an error is detected, `readdir` returns a null pointer. The following `errno` error conditions are defined for this function:
>
> `EBADF`
>
> > The *dirstream* argument is not valid.
>
> To distinguish between an end-of-directory condition or an error, you must set `errno` to zero before calling `readdir`. To avoid entering an infinite loop, you should stop reading from the directory after the first error.

This section describes what you find in a single directory entry, as you might obtain it from a directory stream. All the symbols are declared in the header file `dirent.h`.

Data Type: **`struct dirent`**

This is a structure type used to return information about directory entries. It contains the following fields:

`char d_name[]`

This is the null-terminated file name component. This is the only field you can count on in all POSIX systems.

`ino_t d_fileno`

This is the file serial number. For BSD compatibility, you can also refer to this member as `d_ino`. On GNU/Linux and GNU/Hurd systems and most POSIX systems, for most files this the same as the `st_ino` member that `stat` will return for the file. See File Attributes.

`unsigned char d_namlen`

This is the length of the file name, not including the terminating null character. Its type is `unsigned char` because that is the integer type of the appropriate size. This member is a BSD extension. The symbol `_DIRENT_HAVE_D_NAMLEN` is defined if this member is available.

`unsigned char d_type`

This is the type of the file, possibly unknown. The following constants are defined for its value:

`DT_UNKNOWN`

The type is unknown. Only some filesystems have full support to return the type of the file, others might always return this value.

`DT_REG`

A regular file.

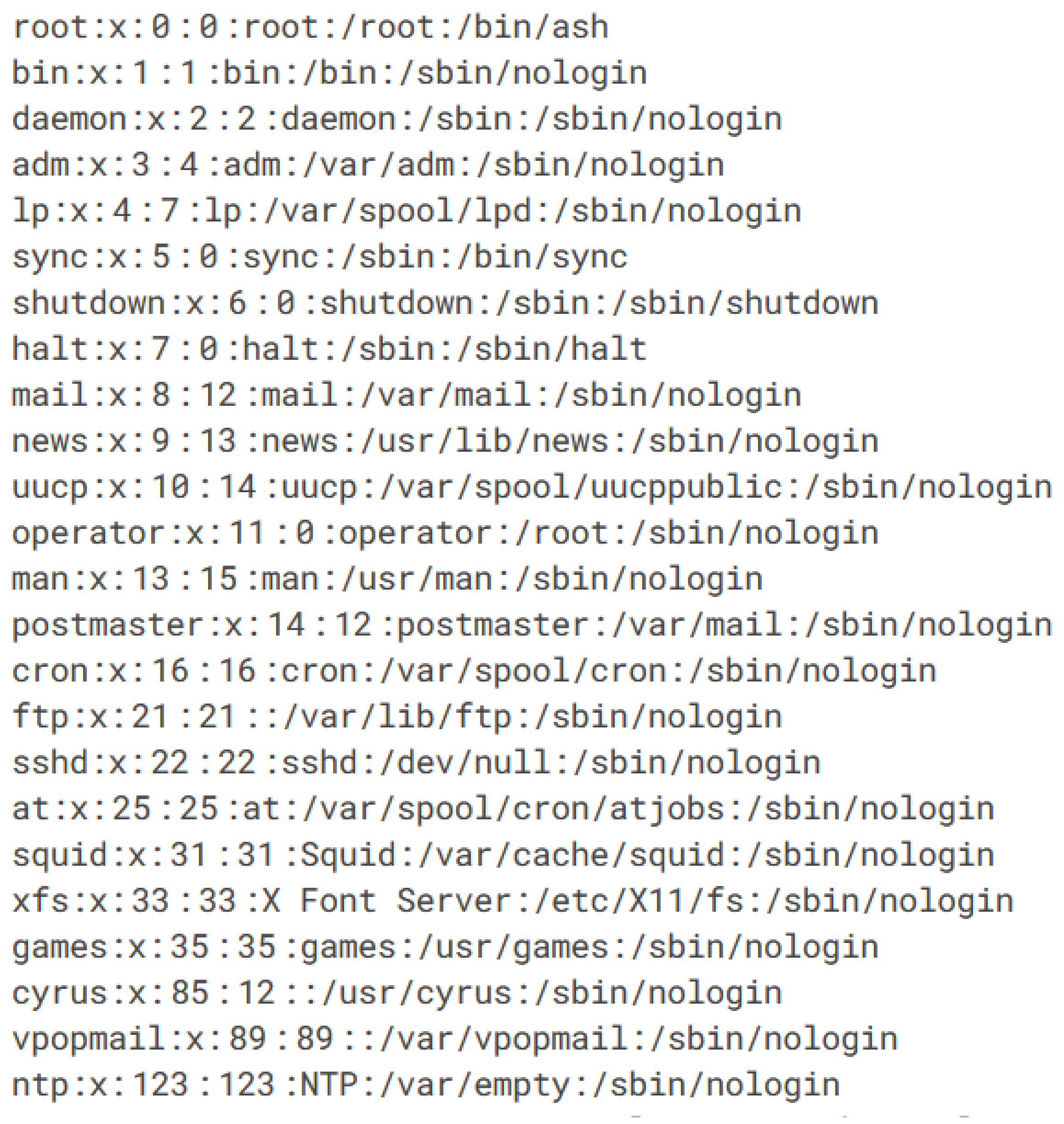
```
int stat(const char *filename, struct stat *buf);
int lstat(const char *filename, struct stat *buf);
int fstat(int filedesc, struct stat *buf);
```

stat structure [edit]

This structure is defined in `sys/stat.h` header file as follows, although implementations are free to define additional fields:[3]

```
struct stat {
    mode_t          st_mode;
    ino_t           st_ino;
    dev_t           st_dev;
    dev_t           st_rdev;
    nlink_t         st_nlink;
    uid_t           st_uid;
    gid_t           st_gid;
    off_t           st_size;
    struct timespec st_atim;
    struct timespec st_mtim;
    struct timespec st_ctim;
    blksize_t       st_blksize;
    blkcnt_t        st_blocks;
};
```

/etc/passwd



```
root:x:0:0:root:/root:/bin/ash
bin:x:1:1:bin:/bin:/sbin/nologin
daemon:x:2:2:daemon:/sbin:/sbin/nologin
adm:x:3:4:adm:/var/adm:/sbin/nologin
lp:x:4:7:lp:/var/spool/lpd:/sbin/nologin
sync:x:5:0:sync:/sbin:/bin/sync
shutdown:x:6:0:shutdown:/sbin:/sbin/shutdown
halt:x:7:0:halt:/sbin:/sbin/halt
mail:x:8:12:mail:/var/mail:/sbin/nologin
news:x:9:13:news:/usr/lib/news:/sbin/nologin
uucp:x:10:14:uucp:/var/spool/uucppublic:/sbin/nologin
operator:x:11:0:operator:/root:/sbin/nologin
man:x:13:15:man:/usr/man:/sbin/nologin
postmaster:x:14:12:postmaster:/var/mail:/sbin/nologin
cron:x:16:16:cron:/var/spool/cron:/sbin/nologin
ftp:x:21:21::/var/lib/ftp:/sbin/nologin
sshd:x:22:22:sshd:/dev/null:/sbin/nologin
at:x:25:25:at:/var/spool/cron/atjobs:/sbin/nologin
squid:x:31:31:Squid:/var/cache/squid:/sbin/nologin
xfs:x:33:33:X Font Server:/etc/X11/fs:/sbin/nologin
games:x:35:35:games:/usr/games:/sbin/nologin
cyrus:x:85:12::/usr/cyrus:/sbin/nologin
vpopmail:x:89:89::/var/vpopmail:/sbin/nologin
ntp:x:123:123:NTP:/var/empty:/sbin/nologin
```