IEEE P1003.2 Draft 11.2 – September 1991

```
head *
```

History of Decisions Made

The `head` utility was not in early drafts. It was felt that `head`, and its frequent companion, `tail`, were useful mostly to interactive users, and not application programs. However, balloting input suggested that these utilities actually do find significant use in scripts, such as to write out portions of log files. Although it is possible to simulate `head` with `sed 10q` for a single file, the working group decided that the popularity of `head` on historical BSD systems warranted its inclusion alongside `tail`.

An earlier draft had the synopsis line:

`head [ −c | −l ] [−n` *number* `] [`*file* . . . `]`

This was changed to the current form based on comments and objections noting that `−c` has not been provided by historical versions of `head` and other utilities in POSIX.2 provide similar functionality. Also, `−l` was changed to `−n` to match a similar change in `tail`.

4.30 `id` — Return user identity

4.30.1 Synopsis

`id [`*user* `]`

`id −G [−n] [`*user* `]`

`id −g [−nr] [`*user* `]`

`id −u [−nr] [`*user* `]`

4.30.2 Description

If no *user* operand is provided, the `id` utility shall write the user and group IDs and the corresponding user and group names of the invoking process to standard output. If the effective and real IDs do not match, both shall be written. If multiple groups are supported by the underlying system (see the description of {NGROUPS_MAX} in POSIX.1 {8}), the supplementary group affiliations of the invoking process also shall be written.

If a *user* operand is provided and the process has the appropriate privileges, the user and group IDs of the selected user shall be written. In this case, effective IDs shall be assumed to be identical to real IDs. If the selected user has more than one allowable group membership listed in the group database (see POSIX.1 {8} section 9.1), these shall be written in the same manner as the supplementary groups described in the preceding paragraph.

4.30.3 Options

The `id` utility shall conform to the utility argument syntax guidelines described in 2.10.2.

The following options shall be supported by the implementation:

`−G` Output all different group IDs (effective, real, and supplementary) only, using the format `"%u\n"`. If there is more than one distinct group affiliation, output each such affiliation, using the format `" %u"`, before the `<newline>` is output.

`−g` Output only the effective group ID, using the format `"%u\n"`.

`−n` Output the name in the format `"%s"` instead of the numeric ID using the format `"%u"`.

`−r` Output the real ID instead of the effective ID.

`−u` Output only the effective user ID, using the format `"%u\n"`.

4.30.4 Operands

The following operand shall be supported by the implementation:

user The login name for which information is to be written.

4.30.5 External Influences

4.30.5.1 Standard Input

None.

4.30.5.2 Input Files

None.

4.30.5.3 Environment Variables

The following environment variables shall affect the execution of `id`:

LANG This variable shall determine the locale to use for the locale categories when both **LC_ALL** and the corresponding environment variable (beginning with **LC_**) do not specify a locale. See 2.6.

LC_ALL This variable shall determine the locale to be used to override any values for locale categories specified by the settings of **LANG** or any environment variables beginning with **LC_**.

LC_CTYPE This variable shall determine the locale for the interpretation of sequences of bytes of text data as characters (e.g., single- versus multibyte characters in arguments).

LC_MESSAGES This variable shall determine the language in which messages should be written.

4.30.5.4 Asynchronous Events

Default.

4.30.6 External Effects

4.30.6.1 Standard Output

The following formats shall be used when the LC_MESSAGES locale category specifies the POSIX Locale. In other locales, the strings `uid`, `gid`, `euid`, `egid`, and `groups` may be replaced with more appropriate strings corresponding to the locale.

`"uid=%u(%s) gid=%u(%s)\n"`, *<real user ID>*, *<user-name>* ,
<real group ID>, *<group-name>*

If the effective and real user IDs do not match, the following shall be inserted immediately before the `\n` character in the previous format:

`" euid=%u(%s)"`,

with the following arguments added at the end of the argument list:

<effective user ID>, *<effective user-name>*

If the effective and real group IDs do not match, the following shall be inserted directly before the `\n` character in the format string (and after any addition resulting from the effective and real user IDs not matching):

`" egid=%u(%s)"`,

with the following arguments added at the end of the argument list:

<effective group-ID>, *<effective group name>*

If the process has supplementary group affiliations or the selected user is allowed 1
to belong to multiple groups, the first shall be added directly before the `<new-` 1
`line>` character in the format string:

`" groups=%u(%s)"`

with the following arguments added at the end of the argument list:

<supplementary group ID>, *<supplementary group name>*

and the necessary number of the following added after that for any remaining supplementary group IDs:

```
",%u(%s)"
```

and the necessary number of the following arguments added at the end of the argument list:

<supplementary group ID>, *<supplementary group name>*

If any of the user ID, group ID, effective user ID, effective group ID, or supplementary/multiple group IDs cannot be mapped by the system into printable user or group names, the corresponding (`%s`) and name argument shall be omitted from the corresponding format string.

When any of the options are specified, the output format shall be as described under 4.30.3.

4.30.6.2 Standard Error

Used only for diagnostic messages.

4.30.6.3 Output Files

None.

4.30.7 Extended Description

None.

4.30.8 Exit Status

The `id` utility shall exit with one of the following values:

- 0 Successful completion.
- >0 An error occurred.

4.30.9 Consequences of Errors

Default.

4.30.10 Rationale. *(This subclause is not a part of P1003.2)*

Examples, Usage

The functionality provided by the 4BSD `groups` utility can be simulated using:

`id -Gn [`*user*`]`

Note that output produced by the `-G` option and by the default case could potentially produce very long lines on systems that support large numbers of supplementary groups. (On systems with user and group IDs that are 32-bit integers and with group names with a maximum of 8 bytes per name, 93 supplementary groups plus distinct effective and real group and user IDs could theoretically overflow the 2048-byte {LINE_MAX} text file line limit on the default output case. It would take about 186 supplementary groups to overflow the 2048-byte barrier using `id −G`.) This is not expected to be a problem in practice, but in cases where it is a concern, applications should consider using `fold −s` (see 4.25) before postprocessing the output of `id`.

History of Decisions Made

The 4BSD command `groups` was considered, but was not used as it did not provide the functionality of the `id` utility of the *SVID*. Also, it was thought that it would be easier to modify `id` to provide the additional functionality necessary to systems with multiple groups than to invent another command.

The options `−u`, `−g`, `−n`, and `−r` were added to ease the use of `id` with shell commands substitution. Without these options it is necessary to use some preprocessor such as `sed` to select the desired piece of information. Since output such as that produced by `id -u -n` is wanted frequently, it seemed desirable to add the options.

4.31 `join` — Relational database operator

4.31.1 Synopsis

`join` [`−a` *file_number* | `−v` *file_number*] [`−e` *string*] [`−o` *list*] [`−t` *char*]
    [`−1` *field*] [`−2` *field*] *file1 file2*

Obsolescent version:

`join` [`−a` *file_number*] [`−e` *string*] [`−j` *field*] [`−j1` *field*] [`−j2` *field*]
    [`−o` *list* ...] [`−t` *char*] *file1 file2*

4.31.2 Description

The `join` utility shall perform an ‘‘equality join’’ on the files *file1* and *file2*. The joined files shall be written to the standard output.

The ‘‘join field’’ is a field in each file on which the files are compared. There shall be one line in the output for each pair of lines in *file1* and *file2* that have identical join fields. The output line by default shall consist of the join field, then the remaining fields from *file1*, then the remaining fields from *file2*. This format can be changed by using the `−o` option (see below). The `−a` option can be used to add unmatched lines to the output. The `−v` option can be used to output only unmatched lines.