CptS 360 (System Programming)
Unit 8: System Data Files

Bob Lewis

School of Engineering and Applied Sciences
Washington State University

Spring, 2022

Motivation

- System programs often need to find things out about their environment:
  - users
  - groups
  - accounting information
  - time and date
- These APIs are good examples for small-scale systems.

Reference

- Stevens & Rago Ch. 6
- relevant *man* pages

Password File

Data in `/etc/passwd` (and LDAP and NIS and ?) now accessed via:

- *getpwuid(3)* and *getpwuid_r(3)*
- *getpwnam(3)* and *getpwnam_r(3)*
- *getpwent(3)* and *getpwent_r(3)*
- *setpwent(3)* and *setpwent_r(3)*
- *endpwent(3)* and *endpwent_r(3)*

Q: What are the "`_r`" variations for?
(Run the `demos/d`*n*`_get_user_data` demo.)
The passwords *themselves* are kept in `/etc/shadow` (or its equivalent. Remember why?), so...

Shadow Password File

Data in `/etc/shadow` (and elsewhere) now accessed by key:

- *getspnam(3)* and *getspnam_r(3)*

or sequentially:

- *setspent(3)* and *setspent_r(3)*
- *getspent(3)* and *getspent_r(3)*
- *endspent(3)* and *endspent_r(3)*

and similar functions.

Group File

Data in `/etc/group` (and elsewhere) now accessed via key:

- *getgrgid(3)* and *getgrgid_r(3)*
- *getgrnam(3)* and *getgrnam_r(3)*

or sequentially:

- *setgrent(3)*
- *getgrent(3)*
- *endgrent(3)*

or for a given user (hint: *whocan*):

- *getgrouplist(3)*

This data does not depend on the calling process.

Supplementary Group IDs

- *getgroups(2)*
  gets supplementary group IDs (non-login groups of process owner) of the calling process
- *setgroups(2)*
  sets supplementary group IDs of the calling process (privilege `CAP_SETGID` to call)
- *initgroups(3)*
  initializes supplementary group IDs for a given user

Other Data "Files"

See Stevens & Rago Figure 6.6 ("Similar Routines ...")
used for

- hosts (e.g. *gethostbyaddr(3)* and *gethostbyaddr_r(3)*)
- networks (e.g. *getnetbyaddr(3)* and *getnetbyaddr_r(3)*)
- protocols (e.g. *getprotobyaddr(3)* and *getprotobyaddr_r(3)*)
- services (e.g. *getservbyaddr(3)* and *getservbyaddr_r(3)*)

Same pattern (`set` – `get` – `end`) for sequential access.

Login Accounting

Two files:

- `/var/run/utmp`
  who's currently logged on
- `/var/log/wtmp`
  history of logins/logouts

Keyed access:

- *getutid(3)*
- *setutline(3)*

Sequential access:

- *getutent(3)*
- *setutent(3)*
- *endutent(3)*

System Identification

- *uname(2)*
- *gethostname(2)*

Kernel Timekeeping

- The epoch is 00:00:00 on Thursday, Jan 1, 1970, UTC. (Earlier dates are negative.)
- Time and date are kept as single quantity.
- kernel functions:
  - *time(2)*
    - returns a `time_t` ("`signed int`") scalar
    - standard
    - allows 1 sec precision
  - *gettimeofday(2)*
    - fills in a "`struct timeval *`"
    - allows μsec precision
    - use `timer*` (e.g. *timeradd(3)* functions to do `timeval` arithmetic
    - timezone support is implementation-dependent

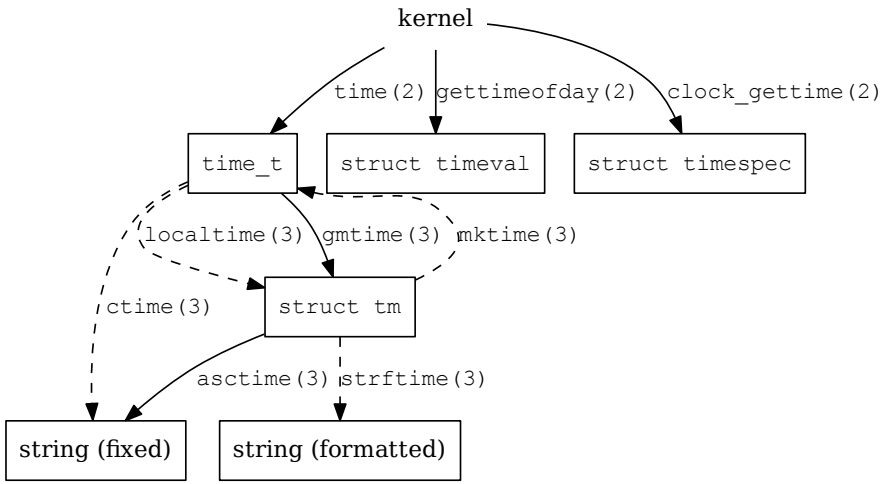
What's so special about 03:14:07 UTC on Tuesday, January 19, 2038? How about Sunday, December 4, 292,277,026,596 (AD)?

Utility Time Functions

- *asctime(3)* and *asctime_r(3)*
- *ctime(3)* and *ctime_r(3)*
  Equivalent to `asctime(localtime(t))`.
- *gmtime(3)* and *gmtime_r(3)*
- *localtime(3)* and *localtime_r(3)*
- *mktime(3)*
- *strftime(3)*
- *strptime(3)*

All work with `time_t`'s.
(Run the `demos/d`*n*`_what_date` demo.)

Timekeeping Function Interdependencies



From Stevens & Rago Figure 6.8 ("Relationship of the various ..."). The dashed conversions depend on the local time zone.