Dates and Times

- Calendar Time Routines
- Calendar Time Routines Continued
- Time Routines Example
- Using `localtime()`
- Date-Time Arithmetic using `mktime()`
- Timezones and Locales
- Process Time
- References

Calendar Time Routines

Calendar Time `time_t`

Seconds since Jan 1, 1970 00:00 UTC. Obtained from kernel using `time()` or `gettimeofday()` (microseconds precision). 32-bit Linux uses signed int for `time_t` with consequent range between 1901 to 2038.

Broken-Down Time `struct tm`

A struct containing different fields for time components like year, month, minute, day, weekday, etc. `gmtime()` and `localtime()` converts `time_t` to broken-down time (`localtime()` adjusts return value for the local-time according to the `TZ` environmental variable). `mktime()` converts a broken-down time to a calendar time.

Calendar Time Routines Continued

String Representation `char *`

`ctime()` converts `time_t` to a local-time string. `asctime()` converts a broken-down time to string; `strftime()` converts broken-down time to a string formatted using a `printf`-like format string. `strptime()` can be used to parse a string into a broken-down time.

Time Routines Example

./programs/calendar-time.c:

```
int main(void)
{
  time_t t = time(NULL);
  printf("time_t t = %ld\n", (long)t);
  printf("ctime(&t): %s", ctime(&t));
  printf("asctime(gmtime(&t)): %s", asctime(gmtime(&t)));
  return 0;
}
```

```
$ ./calendar-time
time_t t = 1462150521
ctime(&t): Sun May  1 20:55:21 2016
asctime(gmtime(&t)): Mon May  2 00:55:21 2016
$
```

Using localtime()

In ./programs/localtime.c:

```
int main() {
  time_t now = time(NULL);
  struct tm *tP = localtime(&now);
  static const char *const months[] = {
    "Jan", "Feb", "Mar", "Apr", "May", "Jun",
    "Jul", "Aug", "Sep", "Oct", "Nov", "Dec",
  };
  printf("using localtime() components: %02d %s %d\n",
         tP->tm_mday, months[tP->tm_mon], tP->tm_year + 1900);
  char buf[100];
  strftime(buf, sizeof(buf), "%d %b %Y", tP);
  printf("using strftime() on localtime: %s\n", buf);
  strftime(buf, sizeof(buf), "%Y-%m-%dT%H:%M:%S", tP);
  printf("ISO-8601 localtime: %s\n", buf);
  strftime(buf, sizeof(buf), "%Y-%m-%dT%H:%M:%S", gmtime(&now));
  printf("ISO-8601 UTC time: %s\n", buf);
  return 0;
}

$ ./localtime
using localtime() components: 01 May 2016
using strftime() on localtime: 01 May 2016
ISO-8601 localtime: 2016-05-01T21:34:01
ISO-8601 UTC time: 2016-05-02T01:34:01
$
```

Date-Time Arithmetic using `mktime()`

Besides converting broken-down time to `time_t`, `mktime()` adjusts broken-down time values appropriately (with `tm_wday` and `tm_yday` taken as output fields):

In ./programs/date-time-arith.c:

```
int main() {
  static const char *const months[] = {
    "Jan", "Feb", "Mar", "Apr", "May", "Jun",
    "Jul", "Aug", "Sep", "Oct", "Nov", "Dec",
  };
  time_t now = time(NULL);
  struct tm *tP = localtime(&now);
  tP->tm_mday -= 365;
  tP->tm_isdst = -1;  //DST?
  time_t t1 = mktime(tP);
  printf("last year at time_t %ld: %02d %s %d\n", t1,
         tP->tm_mday, months[tP->tm_mon], tP->tm_year + 1900);
  return 0;
}

$ ./date-time-arith
last year at time_t 1430617665: 02 May 2015
$
```

Timezones and Locales

- `TZ` and `LOCALE` environmental variables.
- C library takes care of accessing information stored in system files (`/usr/share/zoneinfo`)
- Locale affects case-conversion and ordering rules, formatting for numeric, monetary and date/time values.

Process Time

- Measured in clock-ticks; use `sysconf(_SC_CLK_TCK)` to discover number of clock-ticks per second.
- `clock_t()` returns total CPU time used by process in clock ticks.
- `times()` returns cpu time in user/system space for calling process and waited on children in `struct tms` structure; return value gives elapsed real-time since start of process (wall-clock time). All values in clock ticks `clock_t`;
- POSIX realtime clocks `clock_tgettime()` allow retrieving times in finer resolution (subject to system-clock resolution limits).

References

Text: Ch 10