Time

Advanced Embedded Linux Development

with **Dan Walkes**

University of Colorado **Boulder**

Learning objectives:

Time measurement types and formats

Clock Types

Getting and Setting Time

Linux Time Measurement Types

- Wall time
- Process Time
- Monotonic time

Time Measurement - Wall time

- Actual time and date in the real world (clock)
- Useful for user interaction
  - What time is it now?
  - What time did something happen?
- Localized (timezone, daylight savings)
- May be saved/restored using battery power and hardware support
- Updated with Network Time Protocol (NTP)

Linux Time Measurement Types

- Process Time
  - Time spent executing on a processor
  - Can be less than wall time due to multitasking
  - Can exceed wall time if running on multiple processors
- Monotonic time
  - Always increasing (system uptime)

Linux Time Measurement Formats

- Relative Time
- Absolute Time

Relative Time Measurements

- Relative to some specific time (like now)
- 5 seconds from now, 10 minutes ago
- Typically use monotonic time
  - Changing time reference could be a problem

Absolute Time Measurements

- Not relative to a benchmark
- September 24th 2019 3:00 PM MDT
- Time of an event in a calendar
- Tracked in Linux relative to the epoch
  - Midnight, Jan 1st 1970

Software Clock in Linux

- System Timer instantiated by the kernel.
- Uses a specific interval frequency.
  - Increments the elapsed time by one unit each time the interval ends - “tick” or “jiffy”
  - “jiffies counter” - 64 bit value
  - HZ - frequency of the system timer,
  - platform dependent, historical x86 values range from 100 Hz to 1000 Hz

Software Clock in Linux

- What about jiffies timer rollover?
  - for x86 each jiffy is 0.004 seconds, rollover in 2^64*0.004s = ~5.8 billion years
- Frequency of the system timer is call HZ - architecture specific.
- POSIX time APIs will convert time for us, we don’t need to use HZ.

Time Representations

- time_t in <time.h> is defined as
  typedef long time_t;
- Represents seconds since the epoch (midnight Jan 1st 1970 UTC)
  - 32 bit machine long value will rollover Monday 18th January 2038!
    - Y2Kv2 - 32 bit systems will stop working
  - What about sub second precision?

https://en.wikipedia.org/wiki/Year_2038_problem

POSIX Clock Types

- CLOCK_REALTIME
  - system wide wall clock time
  - settable
- CLOCK_MONOTONIC
  - elapsed time since starting point (ie boot)
  - not settable

POSIX Clock Types

- When should you use CLOCK_MONOTONIC over CLOCK_REALTIME?
  - Don’t use CLOCK_REALTIME when you are measuring intervals which are dependent on system time *not* changing.
  - The system time might be set/change backwards

Time Representations

```
struct timeval {
    time_t      tv_sec;     /* seconds */
    suseconds_t tv_usec;    /* microseconds */
};
```

- timeval (microsecond)
  - get/settimeofday, adjtime
- timespec (nanosecond)
  - clock_gettime, clock_settime, nanosleep, clock_nanosleep

```
struct timespec {
    time_t   tv_sec;        /* seconds */
    long     tv_nsec;       /* nanoseconds */
};
```

Linux System Programming Chapter 11
https://linux.die.net/man/3/clock_gettime

Time Representations - tm

```
struct tm {
    int tm_sec;    /* Seconds (0-60) */
    int tm_min;    /* Minutes (0-59) */
    int tm_hour;   /* Hours (0-23) */
    int tm_mday;   /* Day of the month (1-31) */
    int tm_mon;    /* Month (0-11) */
    int tm_year;   /* Year - 1900 */
    int tm_wday;   /* Day of the week (0-6, Sunday = 0) */
    int tm_yday;   /* Day in the year (0-365, 1 Jan = 0) */
    int tm_isdst;  /* Daylight saving time */
};
```

- human readable time
- asctime, asctime_r, mktime, gmtime, gmtime_r, localtime, localtime_r
  - _r functions are thread safe
  - consider strftime instead of asctime

POSIX clock_gettime

```
NAME
       clock_getres, clock_gettime, clock_settime - clock and time functions

SYNOPSIS
       #include <time.h>

       int clock_getres(clockid_t clk_id, struct timespec *res);

       int clock_gettime(clockid_t clk_id, struct timespec *tp);
```

```
Clock 0 (CLOCK_REALTIME) 1581118597.857295263
Clock 1 (CLOCK_MONOTONIC) 209314.045063020
Clock 2 (CLOCK_PROCESS_CPUTIME_ID) 0.000524091
Clock 3 (CLOCK_THREAD_CPUTIME_ID) 0.000527762
```

```c
/**
* Print the clock time for the clock with id @param id and name
* @param clocktype and return in @param ts
*/
bool print_clock_time( clockid_t id, const char *clocktype, struct timespec *ts )
{
    bool success = false;
    int rc = clock_gettime(id,ts);
    if( rc != 0 ) {
        printf("Error %d (%s) getting clock %d (%s) time\n",
               errno,strerror(errno),id,clocktype);
    } else {
        printf("Clock %d (%s) %ld.%09ld\n",id,clocktype,ts->tv_sec,ts->tv_nsec);
        success = true;
    }
    return success;
}
```

Linux System Programming Chapter 11

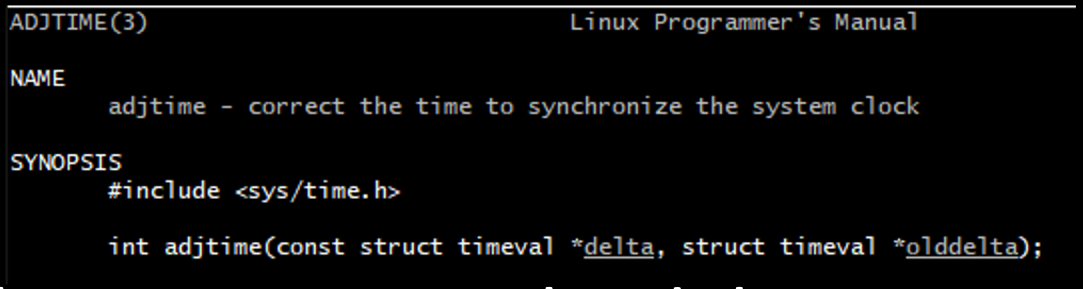
https://github.com/cu-ecen-5013/aesd-lectures/blob/master/lecture9/time_functions.c

Setting Time

```
CLOCK_GETRES(2)                    Linux Programmer's Manual

NAME
       clock_getres, clock_gettime, clock_settime - clock and time functions

SYNOPSIS
       #include <time.h>

       int clock_getres(clockid_t clk_id, struct timespec *res);

       int clock_gettime(clockid_t clk_id, struct timespec *tp);

       int clock_settime(clockid_t clk_id, const struct timespec *tp);
```

- CLOCK_REALTIME is the clk_id to use with these
  - CLOCK_MONOTONIC isn't settable
- May require elevated permissions

Setting Time



```
ADJTIME(3)                          Linux Programmer's Manual

NAME
       adjtime - correct the time to synchronize the system clock

SYNOPSIS
       #include <sys/time.h>

       int adjtime(const struct timeval *delta, struct timeval *olddelta);
```

- Begin adjusting the system time by delta to a more accurate time
  - Slowly adjusts the system clock to match required delta
  - Doesn’t move clock backwards
  - Useful for Network Time Protocol (NTP) daemons