System Programming

Time Management

Seung-Ho Lim

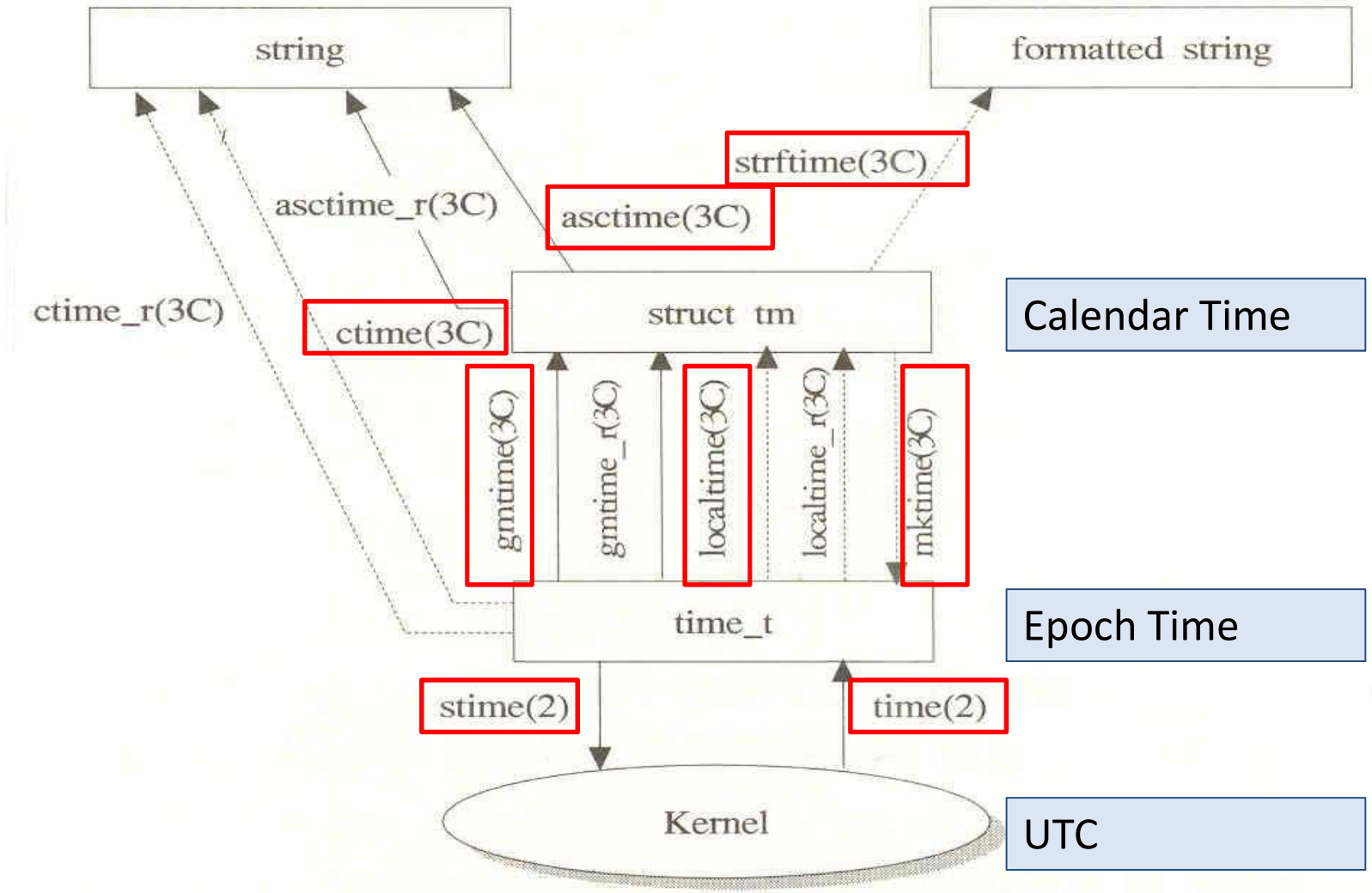
Dept. of Computer & Electronic Systems Eng.

Calendar Time & Process Time

- Calendar time
  - UTC(Coordinated Universal Time) / GMT(Greenwich Mean Time)
    - (in English) Coordinated Universal Time,
    - (in France) Temps Universel Coordonné
  - Epoch time: # of seconds from 1970. 1. 1. 00:00 UTC
  - Data type: *time_t*
- Process/CPU time
  - In clock tick unit
  - 1 tick = 1 ms (Linux) or 10 ms
    - 1 ms = 1/1000 sec
  - *time(1)*: shell command to show seconds to run a program
    - $ time ./a.out

```
real 0m0.002s
user 0m0.000s
sys  0m0.000s
```

Function Relations



time(), stime() : syscall

```
#include <time.h>
time_t time (time_t *tloc);
```

Input

tloc : buffer for time_t (epoch time)

Return

normal : current epoch time, error : -1

```
int stime (const time_t *tp);
```

Input

tp : the epoch time to set

Return

normal : 0, error : -1

get/set time (1)

- setTime.c

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <time.h>

int main( int argc, char *argv[])
{
	time_t curtime, modtime;

	if ( argc != 3) {
		perror("argument error");
		printf("format : [+|-] sec");
		exit(1);
	}
	modtime = (time_t) time(&curtime);

	printf ("current time : %d sec from 1970, 1, 1 00 : 00 : 00\n", (int) curtime);
```

get/set time (2)

```
    if (!strcmp( argv[1], "-"))
            modtime = curtime - atoi(argv[2]);
    else if (!strcmp(argv[1], "+"))
            modtime = curtime + atoi(argv[2]);
    else {
            perror ("argument error");
            exit(2);
    }
    if (stime(&modtime))            {
            perror("stime error");
    }
    printf ("modified time : %d sec from 1970, 1, 1 00 : 00 : 00\n", (int) modtime);
}
```

Results

$ sudo ./setTime + 3600
current time : 1591942445 sec from 1970, 1, 1 00 : 00 : 00
modified time : 1591946045 sec from 1970, 1, 1 00 : 00 : 00

$ sudo ./setTime - 3600
current time : 1591946074 sec from 1970, 1, 1 00 : 00 : 00
modified time : 1591942474 sec from 1970, 1, 1 00 : 00 : 00

tm structure for Calendar Time

```
struct  tm {
    int     tm_sec;         /* seconds (0~59) */
    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 from 0 to 11 : 0 for Jan. */
    int     tm_year;        /* year from 1900 */
    int     tm_wday;        /* day of the week (0~6): 0 for Sunday */
    int     tm_yday;        /* day of the year  (0~365) */
    int     tm_isdst;       /* daylight savings time */
};
```

tm_isdst	*Description*
tm_isdst > 0	In daylight saving time (summer time)
tm_isdst = 0	Not in daylight saving time
tm_isdst < 0	No information

Conversion of Time-format

```
#include <time.h>
struct tm *localtime (const time_t *clock);  // Local time
    Input
        clock : epoch time
    Return : pointer to tm

struct tm *gmtime (const time_t *clock);   // Greenwich mean time
    Input
        clock : epoch time
    Return :  pointer to tm
```

Conversion of Time-format (1)

- timeF.c

```
#include <stdio.h>
#include <sys/types.h>
#include <unistd.h>
#include <time.h>

int main( void)
{
      time_t curtime;
      struct tm *tmbuf;

      if (time(&curtime) < 0) {
            perror("time error");
            exit(1);
      }
```

Conversion of Time-format (2)

```
printf ("Next is result of gmtime()\n\n");

tmbuf = (struct tm *) gmtime(&curtime);

printf ("Year        : %4d \n", tmbuf->tm_year + 1900);
printf ("Month       : %4d \n", tmbuf->tm_mon + 1);
printf ("Day         : %4d \n", tmbuf->tm_mday);
printf ("Hour        : %4d \n", tmbuf->tm_hour);
printf ("Min         : %4d \n", tmbuf->tm_min);
printf ("Sec         : %4d \n", tmbuf->tm_sec);

if (tmbuf->tm_isdst > 0)
        printf ("Summer time is applied. \n");
if (tmbuf->tm_isdst == 0)
        printf ("Summer time is not applied. \n");
if (tmbuf->tm_isdst < 0)
        printf ("there is no information on summer time. \n");
```

Conversion of Time-format (3)

```
    printf ("Next is result of localtime() \n\n");

    tmbuf = ( struct tm *) localtime(&curtime);

    printf ("Year       : %4d \n", tmbuf->tm_year + 1900);
    printf ("Month      : %4d \n", tmbuf->tm_mon + 1);
    printf ("Day        : %4d \n", tmbuf->tm_mday);
    printf ("Hour       : %4d \n", tmbuf->tm_hour);
    printf ("Min        : %4d \n", tmbuf->tm_min);
    printf ("Sec        : %4d \n", tmbuf->tm_sec);

    if (tmbuf->tm_isdst > 0)
        printf ("Summer time is applied. \n");
    if (tmbuf->tm_isdst == 0)
        printf ("Summer time is not applied. \n");
    if (tmbuf->tm_isdst < 0)
        printf ("there is no information on summer time \n");
}
```

Results

```
$./timeF
Next is result of gmtime()
Year        : 2020
Month       :   6
Day         :  12
Hour        :   6
Min:   15
Sec         :  41
Summer time is not applied.

Next is result of localtime()
Year        : 2020
Month       :   6
Day         :  12
Hour        :  15
Min:   15
Sec         :  41
Summer time is not applied.
```

Human readable → UTC

include <time.h>

time_t mktime (struct tm *timeptr);

Input

timeptr : pointer to tm structure

Return

normal : epoch time

error : -1

Human readable → UTC (1)

- toUTC.c

```
#include <stdio.h>
#include <sys/types.h>
#include <unistd.h>
#include <time.h>

int main( void)
{
	time_t curtime, mkdtime;
	struct tm *tmbuf;

	if (time(&curtime) < 0) {
		perror("time error");
		exit(1);
	}
	printf ("curtime time from 1970, 1, 1 00 : 00 : 00 : %d sec \n\n", (int) curtime);
	printf ("we make tm struct and print time information \n");
```

Human readable → UTC (2)

```
    tmbuf = (struct tm *) localtime(&curtime);

    printf ("%4d year", tmbuf->tm_year + 1900);
    printf ("%4d month", tmbuf->tm_mon + 1);
    printf ("%4d day", tmbuf->tm_mday);
    printf ("%4d hour", tmbuf->tm_hour);
    printf ("%4d minute", tmbuf->tm_min);
    printf ("%4d sec\n\n", tmbuf->tm_sec);

    mkdtime = mktime(tmbuf);

    printf( "Next is result of mktime which make second from tm struct \n");

    printf ("mktime result : %d\n", (int) mkdtime);
}
```

Results

$./toUTC
curtime time from 1970, 1, 1 00 : 00 : 00 : 1591942647 sec

we make tm struct and print time information
2020 year 6 month 12 day 15 hour 17 minute 27 sec

Next is result of mktime which make second from tm struct
mktime result : 1591942647

Time to string (1)

```
#include <time.h>
```

char *ctime (const time_t *clock);

Input

clock : epoch time

Return : time string

char *asctime(const struct tm *tm);

Input

tm : pointer to tm structure

Return : time string

Time to string (2)

- timeToStr.c

```
#include <stdio.h>
#include <sys/types.h>
#include <unistd.h>
#include <time.h>

int main( void)
{     time_t curtime, mktime;
      struct tm *tmbuf;

      if( time(&curtime) < 0) {
            perror("time error");
            exit(1);
      }
      printf ("curtime time from Epoch(UTC) : %d secs\n", (int) curtime);
      printf ("Result of time : %s\n", ctime(&curtime));

      tmbuf = (struct tm *) localtime(&curtime);
      printf("Result of asctime : %s\n", asctime(tmbuf));
}
```

Results

```
$ date
Fri Jun 12 15:18:24 KST 2020

$ ./timeToStr
curtime time from Epoch(UTC) : 1591942735 secs
Result of time : Fri Jun 12 15:18:55 2020

Result of asctime : Fri Jun 12 15:18:55 2020
```

strftime(): conversion by format

```
#include <time.h>
size_t  strftime (char *s, size_t maxsize, const char *format, const struct tm
    *timeptr)
```

Input

- s : string buffer
- maxsize : buffer size
- format : conversion format
- timeptr : tm pointer

Return

- normal : # of characters stored in “s”
- error : 0

Formats for strftime() (1)

변환기호	설명	예제
%%	'%'문자를 출력하는데 사용	%
%a	축약된 요일명	Sun
%A	축약하지 않은 요일명	Sunday
%b	축약된 월명	Jan
%B	축약하지 않은 월명	January
%c	날짜와 시간	Sun Jan 09 18 : 33 : 19 1994
%C	date(1)명령이 만들어내는 날짜와 시간	Sun Jan 09 18 : 33 : 19 KST 1994
%d	일(01 에서 31)	09
%D	%m/%d/%y 형태의 날짜	01/09/94

Formats for strftime() (2)

변환기호	설명	예제
%e	일(1에서 31로서 필요한 숫자만 표시)	9
%h	축약된 월명	Jan
%H	시간(00에서 23)	18
%I	시간(01에서 12)	06
%j	한해의 날수(001에서 366)	009
%k	시간(0에서 23)	18
%l	시간(1에서 12)	6
%m	월 번호(01에서 12)	01
%M	분(00에서 59)	33

Formats for strftime() (3)

변환기호	설명	예제
%n	'₩n'과 동일 새로운 행	
%p	AM이나 PM중 하나와 동일	PM
%r	%I : %M : %S [AM|PM]으로 나타낸 시간	06 : 33 : 19 PM
%R	%H : %M으로 나타낸 시간	18 : 33
%S	윤초를 지원하는 초(00에서 61)	19
%t	탭문자와 동일	탭
%T	%H : %M : %S로 나타낸 시간	18 : 33 : 19
%U	한해의 주 번호(00에서 53), 일요일로 시작하는 주만 세어서 나타냄	02
%w	요일 번호(0에서 6까지로 일요일은 0)	0

Formats for strftime() (4)

변환기호	설명	예제
%W	한해의 주번호(00에서 53) 월요일로 시작하는 주만 세어서 나타냄	01
%x	날짜 표시	01/09/94
%X	시간 표시	18 : 33 :19
%y	그 세기의 해표시(00에서 99)	94
%Y	년도 표시	1994
%Z	Time zone 이름	KST

strftime(3) Example (1)

- strFTime.c

```
#include <stdio.h>
#include <time.h>

void main( void)
{
	struct tm time_str;
	char str[80];
	size_t strsize = 80;
	int year, month, day, hour, minute;

	time_str.tm_sec = 0;
	time_str.tm_min = 0;
	time_str.tm_hour = 0;

	printf ("year : ");
	scanf ("%d", &year);
	printf ("month : ");
	scanf ("%d", &month);
```

strftime(3) Example (2)

```
        printf ("day : ");
        scanf ("%d", &day);
        printf (“hour : ");
        scanf ("%d", &hour);
        printf (“minute : ");
        scanf ("%d", &minute);

        time_str.tm_mday = day;
        time_str.tm_mon = month -1;
        time_str.tm_year = year -1900;
        time_str.tm_hour = hour;
        time_str.tm_min = minute;

        if( mktime (&time_str) == -1)
                perror("mktime");
        if (strftime(str, strsize, " %A %b %d %j %U %X %r ", &time_str) <= 0)
                perror("strftime");
        printf ("%s\n", str);
}
```

Results

```
$ ./strFTime
year : 2020
month : 6
day : 12
hour : 15
minute : 23
 Friday Jun 12 164 23 15:23:00 03:23:00 PM
" %A %b %d %j %U %X %r "
```