4주차 3차시 타이머 관리

【학습목표】

1. 리눅스 시스템 타이머에 대해 설명할 수 있다.
2. 리눅스 시스템 타이머 함수를 사용할 수 있다.

학습내용1 : 리눅스 시스템 타이머

; 1970년 1월 1일 0시 0분 0초(UTC)를 기준으로 현재까지 경과한 시간을 초 단위로 저장하고 이를 기준으로 시간 정보 관리

1. 기본 시간 정보 확인

① 초 단위로 현재 시간 정보 얻기 : time(2)

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

time_t time(time_t *tloc);
```

tloc : 검색할 시간 정보를 저장할 주소

```
01  #include <sys/types.h>
02  #include <time.h>
03  #include <stdio.h>
04
05  int main(void) {
06      time_t tt;
07
08      time(&tt);
09      printf("Time(sec) : %d\n", (int)tt);
10
11      return 0;
12  }
```

```
# ex4_16.out
Time(sec) : 1233361205
```

② 마이크로 초 단위로 시간 정보얻기 : gettimeofday(3)

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

int gettimeofday(struct timeval *tp, void *tzp);
int settimeofday(struct timeval *tp, void *tzp);
```

tp : 시간 정보 구조체 주소
tzp : 시간대

* timeval 구조체

```
struct timeval {
    time_t      tv_sec; /* 초 */
    suseconds_t tv_usec; /* 마이크로 초 */
};
```

```
..
04  int main(void) {
05      struct timeval tv;
06
07      gettimeofday(&tv, NULL);
08      printf("Time(sec) : %d\n", (int)tv.tv_sec);
09      printf("Time(micro-sec) : %d\n", (int)tv.tv_usec);
10
11      return 0;
12  }
```

```
# ex4_17.out
Time(sec) : 1233362365
Time(micro-sec) : 670913
```

2. 시간대 정보 : tzset(3)

현재 지역의 시간대로 시간을 설정
이 함수를 호출하면 전역변수 4개에 정보를 설정

```
#include <time.h>

void tzset(void);
```

```
extern time_t timezone, altzone;
extern int daylight;
extern char *tzname[2];
```

timezone : UTC와 지역 시간대와 시차를 초 단위로 저장
altzone : UTC와 일광절약제 등으로 보정된 지역시간대와의 시차를 초 단위로 저장
daylight : 일광절약제를 시행하면 0이 아니고, 아니면 0
tzname : 지역시간대와 보정된 시간대명을 약어로 저장

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

int main(void) {
    tzset();

    printf("Timezone : %d\n", (int)timezone);
    printf("Altzone : %d\n", (int)altzone);
    printf("Daylight : %d\n", daylight);
    printf("TZname[0] : %s\n", tzname[0]);
    printf("TZname[1] : %s\n", tzname[1]);

    return 0;
}
```

```
# ex4_18.out
Timezone : -32400
Altzone : -36000
Daylight : 1
TZname[0] : KST
TZname[1] : KDT
```

UTC와 9시간(32,400초) 시차가 발생

3. 시간의 형태 변환

① 초 단위 시간 정보 분해 : gmtime(3), localtime(3)

```c
#include <time.h>

struct tm *localtime(const time_t *clock);
struct tm *gmtime(const time_t *clock);
```

초를 인자로 받아 tm구조 리턴, gmtime은 UTC기준, localtime은 지역시간대 기준

② 초 단위 시간으로 역산 : mktime(3)

```c
#include <time.h>

time_t mktime(struct tm *timeptr);
```

③ tm구조체

```
struct tm {
    int tm_sec;
    int tm_min;
    int tm_hour;
    int tm_mday;
    int tm_mon;
    int tm_year;
    int tm_wday;
    int tm_yday;
    int tm_isdst;
};
```

tm_mon(월) : 0(1월)~11(12월)
tm_year(연도) : 년도 – 1900
tm_wday(요일) : 0(일)~6(토)
tm_isdst(일광절약제) : 1(시행)

gmtime, localtime 함수 사용하기

```
01 #include <time.h>
02 #include <stdio.h>
03
04  int main(void) {
05      struct tm *tm;
06      time_t t;
07
08      time(&t);
09      printf("Time(sec) : %d\n", (int)t);
10
11      tm = gmtime(&t);
12      printf("GMTIME=Y:%d ", tm->tm_year);
13      printf("M:%d ", tm->tm_mon);
     printf("D:%d ", tm->tm_mday);
15      printf("H:%d ", tm->tm_hour);
16      printf("M:%d ", tm->tm_min);
17      printf("S:%d\n", tm->tm_sec);
18
19      tm = localtime(&t);
20      printf("LOCALTIME=Y:%d ", tm->tm_year);
21      printf("M:%d ", tm->tm_mon);
22      printf("D:%d ", tm->tm_mday);
```

gmtime, localtime 함수 사용하기

```
23      printf("H:%d ", tm->tm_hour);
24      printf("M:%d ", tm->tm_min);
25      printf("S:%d\n", tm->tm_sec);
26
27      return 0;
28  }
```

```
# ex4_19.out
Time(sec) : 1233369331
GMTIME=Y:109 M:0 D:31 H:2 M:35 S:31
LOCALTIME=Y:109 M:0 D:31 H:11 M:35 S:31
```

연도가 109?
어떻게 해석해야하나?

mktime 함수 사용하기

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

int main(void) {
    struct tm tm;
    time_t t;

    time(&t);
    printf("Current Time(sec) : %d\n", (int)t);

    tm.tm_year = 109;
    tm.tm_mon = 11;
    tm.tm_mday = 31;
    tm.tm_hour = 12;
    tm.tm_min = 30;
    tm.tm_sec = 0;

    t = mktime(&tm);
    printf("2009/12/31 12:30:00 Time(sec) : %d\n", (int)t);

    return 0;
}
```

```
# ex4_20.out
Current Time(sec) : 1233370219
2009/12/31 12:30:00 Time(sec) : 1262226600
```

학습내용2 : 리눅스 시스템 타이머 함수

1. 형식 지정 시간 출력

① 초 단위 시간을 변환해 출력하기: ctime(3)

```
#include <time.h>

char *ctime(const time_t *clock);
```

clock : 초 단위 시간을 저장한 주소

```
01  #include <time.h>
02  #include <stdio.h>
03
04  int main(void) {
05      time_t t;
06
07      time(&t);
08
09      printf("Time(sec) : %d\n", (int)t);
10      printf("Time(date) : %s\n", ctime(&t));
11
12      return 0;
13  }
```

```
# ex4_21.out
Time(sec) : 1233370759
Time(date) : Sat Jan 31 11:59:19 2009
```

② tm 구조체 시간을 변환해 출력하기: asctime(3)

```
#include <time.h>

char *asctime(const struct tm *tm);
```

tm : 시간정보를 저장한 tm 구조체 주소

```
01  #include <time.h>
02  #include <stdio.h>
03
04  int main(void) {
05      struct tm *tm;
06      time_t t;
07
08      time(&t);
09      tm = localtime(&t);
10
11      printf("Time(sec) : %d\n", (int)t);
12      printf("Time(date) : %s\n", asctime(tm));
13
14      return 0;
15  }
```

```
# ex4_22.out
Time(sec) : 1233371061
Time(date) : Sat Jan 31 12:04:21 2009
```

③ 출력 형식 기호 사용 : strftime(3)

```
#include <time.h>

size_t strftime(char *restrict s, size_t maxsize,
const char *restrict format, const struct tm *restrict timeptr);
```

s : 출력할 시간 정보를 저장할 배열 주소
maxsize : s의 크기
format : 출력 형식
timeptr : 출력할 시간정보를 저장한 구조체 주소

④ 형식 지정 시간 출력

지정자	기능	지정자	기능
%a	지역 시간대의 요일명 약자	%A	지역 시간대의 요일명
%b	지역 시간대의 월 이름 약자	%B	지역 시간대의 월 이름
%c	지역 시간대에 적합한 날짜와 시간 표현	%C	date 명령의 결과와 같은 형태로 날짜와 시간 표현
%d	날짜(0~31)	%D	날짜(%m/%d/%y)
%e	날짜(0~31). 한 자리 수는 앞에 공백 추가	%F	%Y-%m-%d 형태로 표현
%g	년도(00~99)	%G	년도(0000~9999)
%h	지역 시간대 월 이름 약자	%j	1년 중 일 수(001~365)
%H	24시간 기준 시간(00~23)	%I	12시간 기준 시간(01~12)
%k	24시간 기준 시간(00~23), 한 자리 수는 앞에 공백 추가	%l	12시간 기준 시간(01~12), 한 자리 수는 앞에 공백 추가
%m	월(01~12)	%M	분(00~59)
%p	지역 시간대 a.m, p.m	%r	%p와 함께 12시간 표시
%R	%H:%M 형태로 시간 표시	%T	%H:%M:%S 형태로 시간 표시
%S	초(00~60)		

⑤ 형식 지정 시간 출력[3]

%n	개행	%t	탭 추가
%U	연중 주간 수 표시(00~53)	%V	ISO 8601 표준으로 연중 주간 수 표시(01~53)
%w	요일(0~6, 0을 일요일)	%W	연중 주간 수 표시(00~53), 1월 첫 월요일이 01주, 그 이전은 00주로 표시
%x	지역 시간대에 적합한 날짜 표시	%X	지역 시간대에 적합한 시간 표시
%y	년도(00~99)	%Y	네 자리 수 년도
%z	UTC와의 시차 표시	%Z	시간대명 약자

strftime 함수 사용하기

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

char *output[] = {
    "%x %X",
    "%G%  mW %d  %U  %H:%M",
    "%r"
};

int main(void) {
    struct tm *tm;
    int n;
    time_t t;
    char buf[257];

    time(&t);
    tm = localtime(&t);

    for (n = 0; n < 3; n++) {
       strftime(buf, sizeof(buf), output[n], tm);
       printf("%s = %s\n", output[n], buf);
    }

    return 0;
}
```

```
# ex4-23.out
%x %X = 01/31/09 12:43:12
%G년 %m월 %d일 %U주 %H:%M = 2009년 01월 31일 04주 12:43
%r = 12:43:12 PM
```

【학습정리】

1. 리눅스 시스템 타이머
- 초 단위로 현재 시간 정보 얻기 : time(2)
- 마이크로 초 단위로 시간 정보얻기 : gettimeofday(3)
- 시간대 정보 : tzset(3)
- 초 단위 시간 정보 분해 : gmtime(3), localtime(3)

2. 리눅스 시스템 타이머 함수
- 초 단위 시간을 변환해 출력하기: ctime(3)
- tm 구조체 시간을 변환해 출력하기: asctime(3)
- 출력 형식 기호 사용 : strftime(3)속도 단위

3. 속도 단위

크기	이름	기호
0.1	데시 (deci)	d
$0.01 = 10^{-2}$	센티 (centi)	c
$0.001 = 10^{-3}$	밀리 (milli)	m
$0.000001 = 10^{-6}$	마이크로 (micro)	μ
$0.000000001 = 10^{-9}$	나노 (nano)	n
$0.000000000001 = 10^{-12}$	피코 (pico)	p
$0.000000000000001 = 10^{-15}$	펨토 (femto)	f
$0.000000000000000001 = 10^{-18}$	아토 (atto)	a
$0.000000000000000000001 = 10^{-21}$	젭토 (zepto	z
$0.000000000000000000000001 = 10^{-24}$	욕토 (yocto)	y