

LAB06

시스템 프로그래밍_류은경Prof

2021115744_권구태

1. (hello1.c 수정) “깜박이”는 텍스트 출력

- command line에서 출력 메시지를 입력받아 깜박거리는 메시지로 출력
- 입력 메시지가 없는 경우는 "Hello, World"를 출력
- sleep, refresh 등 사용

```
$ ./ex1 Hi
Hi
$ ./ex1
Hello, World
$
```

[코드 설명]

```
// 2021115744 권구태
#include <stdio.h>
#include <curses.h>
#define LINES 100
void clearly();

int main(int ac, char *av[])
{
    initscr();
    clear();

    if (ac == 1) {
        while (1)
        {
            addstr("Hello, World");
            clearly();
        }
    } else {
        while (1)
        {
            addstr(av[1]);
            clearly();
        }
    }

    endwin();
    return 0;
}
```

```

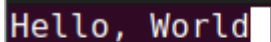
}

void clearly() {
    refresh();
    clear();
    sleep(1);
    refresh();
    clear();
    sleep(1);
}

```

깜빡이는 텍스트를 만들기 위하여, 처음에는 일반적으로 텍스트를 추가하고(**addstr**), 해당 사항을 **refresh** 함수로 터미널에 반영함(출력) 그리고 이를 **clear** 함수로 모두 지우고, 해당 변경 사항을 다시 업데이트 함. 이를 **sleep**를 사용하여 깜빡 거리도록 반영.

[실행 결과]



[주의할 점 및 새롭게 알게 된 사항]

addstr 함수만 사용한다고 해서 터미널에 바로 반영되지 않음. 이는 컴퓨터 내부에서 변경만 일어날 뿐임. 이를 터미널에 적용하고 싶다면 꼭 **refresh** 함수로 적용해줘야 함. **clear** 함수도 마찬가지.

2. (setitimer + sigaction) 매 1초마다 "Tick"을 출력

- 프로그램 종료: Ctrl+C

```

$ ./ex2
Tick
Tick
Tick
...

```

[코드 설명]

```

// 2021115744 권구태
#include <unistd.h> //for pause()
#include <stdio.h>
#include <sys/time.h>
#include <signal.h>
#include <stdlib.h> // for exit()

int set_ticker( int );
int main(void)

```

```

{
    void countdown(int);
    signal (SIGALRM, countdown);
    if ( set_ticker(1000) == -1 )
        perror("set_ticker");
    else
        while( 1 )
            pause();
    return 0;
}
void countdown(int signum)
{
    printf("Tick\n");
    fflush(stdout);
}
int set_ticker( int n_msecs )
{
    struct itimerval new_timeset;
    long    n_sec, n_usecs;

    n_sec = n_msecs / 1000 ;
    n_usecs = ( n_msecs % 1000 ) * 1000L ;
    new_timeset.it_interval.tv_sec = n_sec;
    new_timeset.it_interval.tv_usec = n_usecs;
    new_timeset.it_value.tv_sec = n_sec ;
    new_timeset.it_value.tv_usec = n_usecs ;
    return setitimer(ITIMER_REAL, &new_timeset, NULL);
}

```

- ms 단위에서 s 단위와 ns 단위로 시간을 수정하고(단위만 수정)이를 **setitimer** 함수로 적용함. 적용할 때 구조체를 넘겨줘야 한다.
- **signal** 함수로 **sigalarm**으로 **countdown** 함수를 작동시키고(비동기), **set_ticker** 함수의 인자로 **1000ms(1초)**를 넘겨줌.
- **set_ticker** 함수에서는 1초 단위 타이머를 설정시키고,
- **pause**함수를 while문 내부에 넣음으로써 시그널(countdown)을 무한 반복함.

[실행 결과]

```

명_류은 경 /src/ch07-1$ ./ex2
Tick
Tick
Tick
Tick
Tick
Tick
Tick
Tick
Tick
Tick

```

[주의할 점 및 새롭게 알게 된 사항]

- `setitimer`에 `time`을 적용할 땐 `ms` 단위가 아닌 `s`, `ns` 단위로 수정하여 바꿔야 한다는 점(`itimerval` 구조체가 따로 존재함. 해당 구조체 안에 `s`, `ns`를 지정할 수 있는 변수가 있음).
- `signal`은 비동기로 진행됨.
- `sleep` 함수의 동작 과정
- -> `signal(SIGALRM, handler)` 호출 - `SIGALRM`을 위한 핸들러 등록
- -> `alarm(num seconds)` 호출 - 설정된 초단위로타이머 설정, 프로세스에 `SIGALRM` 시그널을 전송
- -> `pause()` 호출 - 해당 `signal`이 처리될 때까지 프로세스는 일시 중지

3. (타이머 + 사용자 입력 처리)

- 매 1초마다 "Tick: N" 출력: `setitimer` 사용
- 시그널 처리: `sigaction` 사용
- `SIGALRM` 시그널로 타이머 상태를 주기적으로 업데이트
- 사용자 입력 "stop" : 타이머 멈춤
 "resume" : 다시 시작
 "quit" : 종료

```

$ ./ex3
>> Tick: 1
>> Tick: 2
>> stop
Timer stopped.
>> resume
Timer resumed.
>> Tick: 3
>> hello
Unknown command: hello
>> Tick: 4
>> quit
Exiting. Final tick: 4

```

[코드 설명]

```

// 2021115744 권구태
#include <unistd.h>
#include <stdio.h>

```

```

#include <sys/time.h>
#include <signal.h>
#include <stdlib.h>
#include <string.h>
#include <unistd.h>

#define INPUTLEN 100

int set_ticker( int );
void countdown(int signum);
void inthandler(int s);

int count = 0;
int flag = 0;

int main(void)
{
    struct sigaction newhandler;
    sigset_t      oldmask, newmask;
    void          inthandler();
    char          x[INPUTLEN];

    newhandler.sa_handler = inthandler;
    newhandler.sa_flags = SA_RESTART;

    sigemptyset(&newmask);
    sigaddset(&newmask, SIGALRM);

    signal (SIGALRM, countdown);
    if ( set_ticker(1000) == -1 )
        perror("set_ticker");
    else {
        if ( sigaction(SIGINT, &newhandler, NULL) == -1 )
            perror("sigaction");
        else
            while( 1 ) {
                if (flag == 0) {
                    pause();
                }
                fgets(x, INPUTLEN, stdin);
                if (strncmp(x, "quit", 4) == 0) {
                    printf("Exiting. Final tick: %d\n", count);
                    exit(1);
                } else if (strncmp(x, "stop", 4) == 0) {

```

```

        flag = 1;
        printf("Timer stopped.\n");
        sigprocmask(SIG_BLOCK, &newmask, &oldmask);
    } else if (strncmp(x, "resume", 6) == 0) {
        flag = 0;
        printf("Timer resumed.\n");
        sigprocmask(SIG_SETMASK, &oldmask, NULL);
    } else {
        printf("잘못된 값 입력.\n");
    }
}

}

return 0;
}

void countdown(int signum)
{
    count++;
    printf("Tick: %d\n", count);
    fflush(stdout);
}

void inthandler(int s)
{
    printf("Called with signal %d\n", s);
    sleep(s);
    printf("done handling signal %d\n", s);
}

```

- 핸들러에 시그널에 의해 중단된 시스템 호출을 자동으로 재시작 할 수 있도록 SA_RESTART를 설정.
- 백업 set : oldblock, 변경 set : newblock
- 블록킹할 집합을 선언한다. sigemptyset(&newblock), sigaddset(&newblock, SIGALRM)
- signal 호출하고, sigaction도 호출
- stop 입력하면 sigprocmask 함수를 통해 oldmask를 백업 mask로 넣고, newmask로 전환. signal이 block됨. newmask는 SIGALRM을 block하려고 만든 마스크라서.
- 다시 resume을 입력하면 백업에는 NULL을 넣고, 설정 mask에는 기존 백업 mask인 oldmask를 설정(SIG_SETMASK).
- 그리고 pause에서 막히기 때문에 flag 변수를 설정하여 stop 했을 땐 pause를 지나치도록 설정함.

[실행 결과]

```
밍_류은경 /src/ch07-1$ ./ex3
Tick: 1
Tick: 2
stop
Timer stopped.
resume
Timer resumed.
Tick: 3
Tick: 4
Tick: 5
Tick: 6
Tick: 7
Tick: 8
stopTick: 9

Timer stopped.
quit
Exiting. Final tick: 9
```

[주의할 점 및 새롭게 알게된 점]

- sigprocmask 사용 방법
- sigalarm이 block되었을 땐 pause 진행이 안되기 때문에 block시 flag 변수로 pause 실행을 막아야 함.