시스템프로그래밍및실습 중간고사 (2016-2학기)

일시: 2016.12.15(목) 13시~14시10분 (70분)

담당교수 : 권영근, 윤석훈

1. (10점) 아래 명제 각각에 대해 참/거짓을 답하시오. (각 문항에 대해 정답이면 1점, 오답 작성 시 -1점, 합계는 0점 이상)

(1)	SIGKILL과 SIGQUIT은 절대로 무시되지 못하는 두 시그널이다.	F
(2)	vmalloc은 물리적으로 연속적인 메모리 할당을 보장하지 않는다.	T
(3)	리눅스는 메모리 관리, 스케줄러, 디스크 관리 등 주요 기능이 하나의 프로그램으로 구성된 일체형 커널이어서 기능별 독립적인 관리가 상대적으로 수월하다.	F
(4)	한 프로세스 내의 쓰레드들은 독립적으로 스케줄링되어 수행된다.	T
(5)	리눅스 모듈 프로그래밍에서 실수연산은 가능하지 않다.	T
(6)	일단 시스템콜(System Call) 처리 루틴이 시작되었다면 시그널은 어떤 상황에서도 시스템콜 처리루틴 종료될 때까지 프로세스에 전달되지 않는다.	F
(7)	시그널핸들러 내에서는 malloc()과 같은 검증된 C 라이브러리 함수는 모두 안전하게 사용 가능하다.	F
(8)	여러 쓰레드로 이루어져 동작하는 프로그램의 경우에는 데이터 보호나 동기화를 위한 별도의 개발 노력이 필요하다.	T
(9)	유닉스 계열 시스템에서는 스레드간 통신을 위한 메시지 큐와 같은 IPC 방안을 제공하기때문에 Thread간 효율적 데이터 공유가 용이하다.	T
(10)	모듈프로그래밍으로 디바이스드라이버 개발하고 커널에 등록했다고 하자. 응용프로그램을 통해서 해당 디바이스드라이버 코드를 실행하기 위해서는 관리자 또는 수퍼유저 권한이 필요하지 않다.	F

2. (12점) 다음 프로그램에 대해서 각각의 질문에 대해 출력결과를 쓰거나 실행결과를 설명하시오. (단, pid는 1234이며 시그널 번호는 아래 표와 같다고 가정함)

SIGINT	2	SIGQUIT	3
SIGKILL	9	SIGUSR1	10
SIGTERM	15	SIGTSTP	20

```
#include  <stdio.h>
#include  <signal.h>
void catchsig(int signo) {
  printf("CATCH signo=%d₩n", signo);
}
int main() {
  sigset_t newmask, oldmask;
  sigemptyset(&newmask);
  sigaddset(&newmask, SIGTSTP);
  sigprocmask(SIG_BLOCK, &newmask, &oldmask);
  signal(SIGKILL, catchsig);
  signal(SIGQUIT, catchsig);
  signal(SIGTSTP, catchsig);
  signal(SIGTERM, catchsig);
  signal(SIGINT, SIG_IGN);
  signal(SIGUSR1, SIG_DFL);
  for (;;) pause();
}
```

(1) 프로그램 실행 후, 쉘에서 "Ctrl+ \ "입력한 경우

(2) 프로그램 실행 후, 쉘에서 "Ctrl+C"입력한 경우

(3) 프로그램 실행 후, 쉘에서 "Ctrl+Z"입력한 경우

(4) 프로그램 실행 후, 다른 쉘에서 "kill 1234"입력한 경우

(5) 프로그램 실행 후, 다른 쉘에서 "kill –9 1234"입력한 경우

(6) 프로그램 실행 후, 다른 쉘에서 "kill –USR1 1234"입력한 경우

3.(12점) 다음 프로그램은 fork()를 이용하는 프로세스 간 시그널 전송 예제 프로그램이다. 프로그램이 에러 없이 실행되고 예외적인 경쟁조건이 발생하지 않는 다고 가정하자. 아래와 같은 화면 출력을 생성할 수 있도록 코드를 완성하시오. (a)-(b)는 한줄 코드임. (주의: 다음과 같은 동작이 순서대로 발행하여야함. 자식은 부모에게 SIGUSR1 신호 전송, 부모는 자식에게 SIGUSR2 전송, 자식 종료, 부모 종료)

```
//Header 파일 include 부분 생략
int c_pid;
static void sig_handler(int signo)    /* interrupts pause() */
{
   switch(signo)
   {
      case SIGUSR1:
         printf("SIGUSR1 signal received from child₩n");
         _____(a)_____
         break;
      case SIGUSR2:
         _____(b)_____
         exit(0);
      default:  break;
   }
}
int main()
{
   c_pid = fork();
   int status;

   signal(SIGUSR1, sig_handler);
   signal(SIGUSR2, sig_handler);

   if (c_pid == 0) {
      sleep(1);
      _____(c)_____
      pause();
   } else {
      _____(d)_____
      wait(&status);
      printf("child has been terminated, so parent terminates ₩n");
   }
}
```

화면출력:

SIGUSR1 signal received from child

SIGUSR2 signal received from parent

child has been terminated, so parent terminates

(a)

(b)

(c)

(d)

4.(10점) POSIX Thread에서는 동기화를 위해 Mutex와 Condition 변수를 제공한다. Mutex와 Condition 변수의 주요 사용목적을 기술하시오.

Mutex:

Condition 변수: 특정조건을 만족시킬때 까지 쓰레드를 대기시킨다.

5. (12점) 다음 프로그램이 실행 가능하면 출력 결과를 쓰고, 실행 가능하지 않다면 그 이유를 쓰시오.

```
#include <stdio.h>
#include <stdlib.h>
#include <pthread.h>
pthread_t    tid1, tid2;

void * thr_fn2(void *arg) {
     void *tret;
     int val = (int)arg + 10;
     printf("added by 10₩n");
     pthread_exit((void *)val);
}
void * thr_fn1(void *arg) {
     void *tret;
     int val = (int)arg * 10;
     printf("multiplied by 10₩n");
     pthread_create(&tid2, NULL, thr_fn2, (void *)val);
     pthread_join(tid2, &tret);
     pthread_exit((void *)tret);
}
int main(void) {
     int           val = 1;
     void          *tret;
     pthread_create(&tid1, NULL, thr_fn1, (void *)val );
     pthread_join(tid1, &tret);
     printf("Result is %d₩n", (int)tret);
     exit(0);
}
```

출력결과: multiplied by 10
added by 10
Result is 20

6.(10점)하나의 프로그램이 여러 Thread로 구성되어 있을 때 어떤 정보는 Thread 별로 유지하며 어떤 정보(데이터)는 Thread 간 공유한다. 별도 유지 정보와 공유 정보를 각각 3가지씩 나열하시오.

(1) 각 Thread 별도유지 정보	(2) Thread 간 공유 정보
Register value Stack Scheduler priority Thread ID a signal mask	Global data Text section heap File descriptor

7.(10점) 다음 프로그램에서 변수 shard_data는 공유변수라고 하자. 공유변수를 보호하기 위하여 (a)와 (b)에 적절한 1줄 코드를 각각 작성하시오.

(참고: pthread_mutex_init(), pthread_mutex_destroy(); pthread_mutex_lock(), pthread_mutex_unlock(), pthread_cond_init(), pthread_cond_destroy(), pthread_cond_wait(), pthread_cond_signal())

```
...생략 ...

pthread_mutex_t qlock = PTHREAD_MUTEX_INITIALIZER;

int update_shared_data(int a)
{
... 생략 ...
          (a)
    shared_data = a;
          (b)

... 생략 ...
}
```

(a) pthread_mutex_lock(&qlock)

(b) pthread_mutex_unlock(&qlock)

8. (12점) 모듈 프로그래밍에 관한 다음 질문에 답하시오. (단, 커널 버전 2.6 가정함.)

(1) 현재 설치되어 있는 모듈들의 목록을 출력하는 명령어를 쓰시오. lsmod

(2) hello.c를 성공적으로 컴파일한 후 모듈을 커널에 삽입하도록 명령어를 쓰시오. sudo insmod hello.ko

(3) (2)번 모듈을 커널에서 제거하도록 명령어를 쓰시오. sudo rmmod hello

(4) 커널 내 export되고 있는 모든 심볼들의 목록이 포함된 proc 파일 시스템의 파일명을 쓰시오. /proc/ksyms

9. (12점) 다음은 문자 디바이스 드라이버를 생성하고 응용하는 프로그램이다. 주석을 참고하여 다음 빈 칸을 작성하시오.

```
static char *buffer = NULL;
int char_open(struct inode *inode, struct file *filp) { return 0; }
int char_release(struct inode *inode, struct file *filp) { return 0; }
ssize_t char_read(struct file *filp, char *buf, size_t count, loff_t *f_pos){
  //커널내 buffer를 사용자영역 buf로 count만큼 복사
  copy_to_user ( ____(1)____ , ____(2)____, count);
  return count;
}
ssize_t char_write(struct file *filp, const char *buf, size_t count, loff_t *f_pos){
  //사용자영역 buf를 커널내 buffer로 count만큼 복사
  ____(3)____ ( ____buffer____ , ____buf____ , count);
  return count;
}
struct file_operations vd_fops={
     .open = char_open,
     .release = char_release,
     .read = char_read,
     .write = char_write
};
int __init virtual_device_init(void){
  ____(6)____ (250, "virtual_device", ____&vd_fops____); //디바이스 등록
  buffer = (char*) ____(8)____ (1024, GFP_KERNEL);
  if(buffer != NULL) memset(buffer, 0, 1024); //할당된 영역 0으로 초기화
  return 0;
}
void __exit virtual_device_exit(void){
  ____(9)____ (250, "virtual_device") ; //등록한 디바이스 제거(해제)
  ____(10)____ (buffer);
}
____(11)____ (virtual_device_init);
____(12)____ (virtual_device_exit);
MODULE_LICENSE("GPL");
```

(1) copy_from_user (2) buffer (3) buf
(4) register_chrdev (5) &vd_fops (6) kmalloc
(7) unregister_chrdev (8) kfree (9) buffer
(10) module_init (11) module_exit (12)