시스템프로그래밍및실습 기말고사 (2017-2학기) 일시: 2017.12.19(화) 15시~16시10분 (70분) 담당교수 : 권영근

학과: _____ 학년: _____ 학번: _____ 성명: _____

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

(1) init프로세스는 1번 프로세스로서 커널의 일부이다.	T
(2) vfork()의 경우 자식 프로세스가 항상 먼저 실행된다.	T
(3) 프로세스 종료를 수행하는 시스템콜은 exit()이다.	T
(4) 프로세스가 자기 자신에게는 시그널을 보낼 수 없다.	F
(5) 쓰레드 id는 시스템 내에서 유일한 값이 아니다.	F
(6) 한 프로세스 내의 쓰레드들은 독립적으로 스케줄링되어 수행된다.	T
(7) kmalloc은 사용자 공간내 연속적인 메모리 할당을 시도하는 함수이다.	T
(8) 새로 삽입한 모듈을 사용하기 위해서는 재부팅을 한다.	T?
(9) 리눅스 모듈 프로그래밍에서 실수연산은 가능하지 않다.	T
(10) 시그널 mask는 쓰레드 사이에 공유된다.	F

2. (10점) 아래 각 빈 칸에 적절한 내용을 채우시오. (각 2점)

(1) 프로세스 id를 구하는 함수는 (getpid)이다.

(2) 프로세스가 종료되었으나 부모프로세스가 종료결과를 가져가지 못한 경우 (좀비) 프로세스라고 한다.

(3) 6가지 exec 계열 함수 중 시스템콜은 (execve)이다.

(4) 무시할 수 없는 두 시그널은 (SIGKILL)과 (SIGSTOP)이다.

(5) 타이머를 설정하는 시스템콜은 (alarm) 이다.

(6) 공유 자원을 접근하기 전에 획득해야 할 lock을 (Mutex)이라 부른다.

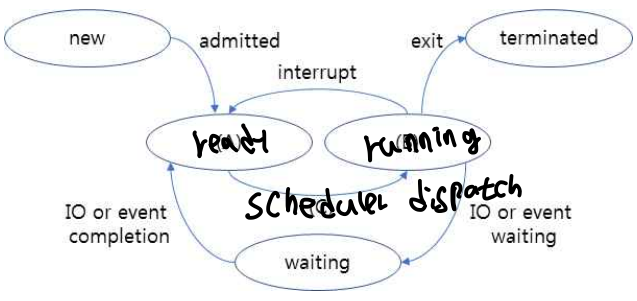
(7) 리눅스와 같이 메모리 관리, 스케줄러, 디스크 관리 등 커널의 주요 기능이 하나의 프로그램으로 구성된 커널을 (Monolithic kernel)이라 한다.

(8) 리눅스에서 하드웨어를 추상화하여 구현하기 만든 인터페이스는 (커널)이다.

(9) 리눅스 디바이스 드라이버 모듈 프로그래밍에서 여러 명령어를 하나의 함수에서 처리할 수 있는데 이를 (vfs_fops) 함수라고 부른다.

(10) 커널프로그램에서 표준출력하기 위해 사용하는 함수는 (printk)이다.

3. (9점) 다음 그림은 프로세스의 상태전이를 나타낸다. 빈 칸 (A)-(C)에 알맞은 내용을 채우시오.



(A) ready (B) running

(C) scheduler dispatch

3. (9점) 아래 프로그램이 오류없이 실행될 때 출력결과를 쓰시오.

```
int g = 60;
int main(void) {
  pid_t pid;
  int v = 10, s = 0;
  pid = vfork();
  if (pid == 0) {
    pid = fork();
    if (pid > 0) {
      g += 10,  v +=30;
      waitpid(pid, &s, 0);
      printf("g:%d, v:%d, s:%d₩n", g, v, WEXITSTATUS(s));
      exit(100);
    }
    g *= 2,  v *= 3;
    printf("g:%d, v:%d, s:%d₩n", g, v, WEXITSTATUS(s));
    exit(200);
  }
  waitpid(pid, &s, 0);
  g += 1,  v += 2;
  printf("g:%d, v:%d, s:%d₩n", g, v, WEXITSTATUS(s));
  exit(3);
}
```

120 30 0
70 40 200
71 42 100

4. (8점) 다음 system() 함수 구현 중 빈 칸에 알맞은 내용을 쓰시오.

```
int system(const char *cmdstring) {
    pid_t    pid;
    int      status;
    if (cmdstring == NULL) return(1);
    if ((pid = ① fork() ) < 0) { status = -1; }
    else if (pid == 0) {                          /* child */
        ② execl "/bin/sh", "sh", "-c", cmdstring, (char *)0);
        _exit(127);
    } else {                                      /* parent */
        while (③ waitpid (pid, &status, 0) < 0)
            if (errno != EINTR) { status = -1; break; }
    }
    return( ④ status );
}
```

① fork() ② execl

③ waitpid ④ status

5. (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

6. (12점) 다음 프로그램을 실행한 후 각각의 상황에 대해 출력결과를 쓰시오. 만약, 프로그램에 아무 변화가 없는 경우에는 "no output", 프로그램이 그냥 종료하는 경우 "terminated"라고 쓰시오. (단, pid는 1234이며 시그널 번호는 아래 표와 같다고 가정함)

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

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

(1) 쉘에서 "Ctrl+C"입력한 경우

(2) 쉘에서 "Ctrl+ \ "입력한 경우

(3) 쉘에서 "Ctrl+Z"입력한 경우

(4) 다른 쉘에서 "kill 1234"입력한 경우

(5) 다른 쉘에서 "kill –9 1234"입력한 경우

(6) 다른 쉘에서 "kill –USR1 1234"입력한 경우

7. (10점) 다음 프로그램의 실행 결과를 쓰시오.

```
pthread_t tid1, tid2;
int glob=2;
void * thr_fn2(void *arg) {
    void *tret;
    int val = (int)arg + 10;
    glob *= 2;
    return ((void *)val);
}
void * thr_fn1(void *arg) {
    void *tret;
    int val = (int)arg * 10;
    glob += 1;
    pthread_create(&tid2, NULL, thr_fn2, (void *)val);
    pthread_join(tid2, &tret);
    pthread_exit((void *)tret);
}
int main(void) {
    int val = 2;
    void *tret;
    pthread_create(&tid1, NULL, thr_fn1, (void *)val );
    pthread_join(tid1, &tret);
    printf("Result is %d %d₩n", (int)tret, glob);
    exit(0);
}
```

8. (8점) 다음은 queue에 메시지를 삽입하는 쓰레드와 메시지를 처리하는 쓰레드가 작동하는 프로그램이다. queue가 비어 있지 않는 경우에만 메시지 처리가 가능하도록 조건 변수를 사용하여 동기화하려고 할 때 빈 칸에 알맞은 함수 이름을 쓰시오.

```
struct msg { struct msg *m_next; /* ... more stuff here ... */ };
struct msg *workq;
pthread_cond_t qready = PTHREAD_COND_INITIALZER;
pthread_mutex_t qlock = PTHREAD_MUTEX_INITIALZER;
void process_msg(void) {
  struct msg *mp;
  for (;;) {
    _____①_____(&qlock);
    while (workq == NULL) _____②_____(&qready, &qlock);
    mp = workq;
    workq = mp->m_next;
    _____③_____(&qlock);
  }
};
void enqueue_msg(struct msg *mp) {
    _____①_____(&qlock);
  mp->m_next = workq;
  workq = mp;
    _____③_____(&qlock);
    _____④_____(&qready);
}
```

① ② ③ ④

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

```
static char *buffer = NULL;
int ch_open(struct inode *i, struct file *f) { return 0; }
int ch_release(struct inode *i, struct file *f) { return 0; }
ssize_t ch_read(struct file *f, 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 ch_write(struct file *f, char *buf, size_t count, loff_t *f_pos){
  //사용자영역 buf를 커널내 buffer로 count만큼 복사
  ___(3)___ ( ___(4)___ , ___(5)___ , count);
  return count;
}
struct file_operations vd_fops={
    .open = ch_open,
    .release = ch_release,
    .read = ch_read,
    .write = ch_write
};
int __init virtual_device_init(void) {
  ___(6)___ (250, "virtual_device", ___(7)___ ); //디바이스 등록
  buffer = (char*) ___(8)___(1024, GFP_KERNEL);
  if (buffer != NULL) memset(buffer, 0, 1024); //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);
```

(1) (2) (3)

(4) (5) (6)

(7) (8) (9)

(10) (11) (12)