

Lab 13

2024학년도 2학기 리눅스시스템

조교 이지원

lee.jiwon@sookmyung.ac.kr



SOOKMYUNG WOMEN'S UNIVERSITY



Lab 13. 프로세스 실습

1. 프로세스 생성

- 다음 프로그램을 실행하고 그 결과를 설명하시오.

```
1 #include <stdio.h>
2 #include <unistd.h>
3
4 /* 자식 프로세스를 생성한다. */
5 int main()
6 {
7     int pid;
8     printf("[%d] 프로세스 시작 \n", getpid());
9     pid = fork();
10    printf("[%d] 프로세스 : 반환값 %d\n", getpid(), pid);
11    pid = fork();
12    printf("[%d] 프로세스 : 반환값 %d\n", getpid(), pid);
13 }
```

1. 프로세스 생성

- 생성된 프로세스는 몇 개인가?
- 그들 사이의 관계는 무엇인가?
- `fork()`를 n 번 하면 몇 개의 프로세스가 생성되는가?

2. 쉘 인터프리터 작성

- 프로그램 13.5를 참고하여 쉘 인터프리터를 작성하시오.
 - 프로그램 13.5는 자식 프로세스를 생성하여 무조건 echo 명령어를 실행한다.
 - echo 명령어 대신에 입력 받은 명령어를 실행하게 한다.
- 쉘 인터프리터의 명령어 실행
 1. 명령어 전면 실행
[shell] 명령어
 2. 명령어 후면 실행
[shell] 명령어 &

[참고] 프로그램 13.5

```
exec2.c
~/linux/chap13

열기(O)  [+]
```

```
1 #include <stdio.h>
2 #include <stdlib.h>
3 #include <unistd.h>
4 #include <sys/wait.h>
5
6 /* 자식 프로세스를 생성하여 echo 명령어를 실행한다 */
7 int main() {
8     int pid, child, status;
9     printf("부모 프로세스 시작\n");
10    pid = fork();
11
12    if (pid == 0) {
13        execl("/bin/echo", "echo", "hello", NULL);
14        fprintf(stderr, "첫 번째 실패");
15        exit(1);
16    }
17    else {
18        child = wait(&status);
19        printf("자식 프로세스 %d 끝\n", child);
20        printf("부모 프로세스 끝\n");
21    }
22    return 0;
23 }
```

```
u2300000@ubuntu-virtualbox:~/linux/chap13$ gcc -o exec2 exec2.c
u2300000@ubuntu-virtualbox:~/linux/chap13$ ./exec2
부모 프로세스 시작
hello
자식 프로세스 2307 끝
부모 프로세스 끝
```

2. 셸 인터프리터 작성 (shell.c)

- 이 인터프리터를 구현하기 위해서 다음과 같이 진행한다.
 - `strtok_r()` 함수
 - `execvp()` 시스템 호출의 기능에 대해 조사한다. (구글링!)
- 셸 인터프리터는 다음과 같이 동작한다.
 - 1) 프롬프트를 내주고 명령어를 입력으로 받는다.
 - 2) `strtok_r()` 함수를 이용하여 이를 명령줄 인수로 분리한다.
 - 3) 분리된 명령줄 인수를 다음 배열에 순차적으로 저장한다.
 - `char *args[MAXARG];`
 - 4) 자식 프로세스를 생성하여 자식 프로세스로 하여금 `execvp()` 시스템 호출을 이용하여 명령어를 실행하게 한다.
 - 5) 부모 프로세스는 자식 프로세스가 끝나기를 기다린다.
 - 6) (1)로 돌아가서 같은 과정을 반복한다.
(후면 실행의 경우에는 기다리지 않고 (1)로 돌아간다.)

2. 쉘 인터프리터 실행 결과 (shell.c)

- 그동안 사용했던 명령어들이 정상적으로 동작해야 한다. (복합 명령어, fg, bg, jobs 제외)

```
u2300000@ubuntu-virtualbox:~/linux/chap13$ ./shell
[shell] echo Hello World!
Hello World!
[shell] date
2024. 12. 03. (화) 13:28:27 KST
[shell] who
u2300000 tty2          2024-12-03 11:48 (tty2)
[shell] ps
  PID TTY          TIME CMD
 1859 pts/0        00:00:00 bash
 4416 pts/0        00:00:00 shell
 4420 pts/0        00:00:00 ps
[shell] cat test.txt
안녕하세요.
리눅스 실습 13번째 시간입니다.
[shell] mkdir hello
[shell] ls
exec2  exec2.c  hello  sample.c  shell  shell.c  test.txt
[shell] cat -n
hello
      1  hello
[shell] quit
```


quit 입력 시 쉘 인터프리터 종료되도록 구현

2. 쉘 인터프리터 실행 결과 (shell.c)

- 명령어 후면 실행 중일 때에는 다른 명령어 실행이 가능하다.

```
u2300000@ubuntu-virtualbox:~/linux/chap13$ ./shell
[shell] sleep 10
[shell] sleep 10 &
[1] 4433
[shell] date
2024. 12. 03. (화) 13:34:00 KST
[shell] ps
  PID TTY          TIME CMD
 1859 pts/0        00:00:00 bash
 4431 pts/0        00:00:00 shell
 4433 pts/0        00:00:00 sleep
 4435 pts/0        00:00:00 ps
[shell]
```

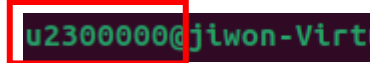
전면처리와 후면처리의 실행 결과를 비교하여 보고서에 설명 작성할 것.



Lab 13. 과제 설명

과제 제출

■ 과제 제출 양식1 (보고서)

1. 자신의 학번으로 된 계정으로 로그인하여 실습을 진행한 후 캡처 
2. 각 문제별 실행 화면 캡처 및 설명을 보고서로 작성
3. 보고서에 실습번호, 분반, 학과, 학번, 이름을 적을 것 (보고서 기본 형식 유지)
4. 워드로 보고서를 작성한 뒤 PDF로 변환하여 제출
5. 파일명: 실습번호_분반_학번_이름 (Lab13_001_2300000_눈송이.pdf)

■ 과제 제출 양식2 (shell.c)

1. 파일명 준수 (미준수 시 0점)
2. 파일에 주석으로 분반, 학과, 학번, 이름 반드시 적을 것

```
// 001 컴퓨터과학과 2300000 눈송이
```

■ 최종 과제 제출 양식 (Lab13_학번.zip)

1. pdf 파일(보고서) + shell.c를 압축한 하나의 zip파일

과제 제출

- 과제 제출 방법
 - Snowboard Lab13에 pdf 파일(보고서) + shell.c를 압축한 하나의 zip파일 제출
 - 제출 전 양식 준수 여부 반드시 확인!!!!!!
 - 제출 기한 : 12월 15일 일요일 23:59:00 (추가 제출 없음)

+) 과제 질문 방법

- lee.jiwon@sookmyung.ac.kr (조교 메일로 질문 보내기)
- 질문 답변 시간: 월-목 10:00-17:00
- 질문 시 주의사항

- 충분히 고민 후 질문 (질문하기 전 구글링 필수!)
- 메일에 반드시 과목, 분반, 전공, 학번, 이름 명시
- 몇 번 과제에서 어떤 부분이 막혔는지, 어떤 과정이 문제인지 **명확한 설명 첨부**
- 코드 질문 시 화면 캡처가 아닌 실제 코드 파일 첨부 (출력 결과는 캡처 가능)
- 답장이 늦을 수 있으니 여유 있게 미리 질문 (특히 과제 제출 마지막 날 유의!)