상명대학교 컴퓨터과학과 유닉스 프로그래밍 실습 지침서

- 실습 번호: lab-06
- 실습 디렉토리: ~/unix/lab-06 이름: _____
- 실습 날짜: 년 월 일
- 실습 제목: Signal and mmap
- 실습 내용: 학번: _____

1. (6-1.c, 6-1) 다음과 같이 파일 6-1.c를 편집하고 6-1로 컴파일하여 수행하시오.

```
#include <sys/types.h>
#include <unistd.h>
#include <signal.h>
#include <stdio.h>

int main(void) {
    printf("Before SIGCONT Signal to parent.\n");

    kill(getppid(), SIGCONT);

    printf("Before SIGQUIT Signal to me.\n");

    kill(getpid(), SIGQUIT);

    printf("After SIGQUIT Signal.\n");

    return 0;
}
```

수행 결과를 적고, 결과를 설명하시오.

2. (6-2.c, 6-2) 아래 프로그램을 파일 6-2.c에 입력하고 컴파일하여 6-2를 만드시오.

```
int main(void)
{
  int i;

  for (i = 0; i < 20; i++) {
    printf("sleep 1 second...\n");
    sleep(1);
  }
  return 0;
}
```

컴파일된 프로그램을 수행시킨 후 프로그램이 종료하기 전에 키 보드에서 '^C'를 입력해보시오. 이때 어떤 현상이 일어나는 지를 관찰하여 기록하고 그 이유를 설명하시오.

3. (6-3.c, 6-3) "6-2.c"를 수정하여 이 프로그램이 수행되는 도중 키보드에서 '^C'를 입력하더라도 이를 무시하고 계속 수행되게 하는 프로그램 "6-3.c"를 만드시오. 프로그램을 컴파일 후 수행시켜 원하는 동작이 일어나는 지를 살펴보고 그 결과와 작동 원리를 적으시오.

4. (6-4.c, 6-4) "6-3.c"를 수정하여 이 프로그램이 수행되는 도중 키 보드에서 '^C'를 입력하면 표준 출력에 "catch SIGINT..."을 출력하고 프로그램을 계속 수행시키는 프로그램 "6-4.c"를 작성하시오. 프로그램을 컴파일 후 수행시켜 원하는 동작이 일어나는 지를 살펴보고 그 결과와 작동 원리를 적으시오.

5. (6-5.c, 6-5) 다음과 같은 프로그램 "6-5.c"를 만들고 컴파일하여 실행하시오.

```
#include <unistd.h>
#include <signal.h>
#include <siginfo.h>
#include <stdio.h>

void handler(int signo) {
    psignal(signo, "Received Signal");
}

int main(void) {
    sigset(SIGALRM, handler);

    alarm(2);
    printf("Wait...\n");
    sleep(10);

    return 0;
}
```

실행 결과를 적고 왜 그런 결과가 나오는지를 설명하시오.

6. (6-6.c, 6-6) 다음과 같이 파일 6-6.c를 편집하고 6-6으로 컴파일하여 수행하시오.

```
#include <sys/types.h>
#include <sys/mman.h>
#include <sys/stat.h>
#include <fcntl.h>
#include <unistd.h>
#include <stdlib.h>
#include <stdio.h>

int main(intargc, char *argv[]) {
intfd;
pid_tpid;
caddr_taddr;
struct stat statbuf;

if (argc != 2) {
fprintf(stderr, "Usage : %s filename\n", argv[0]);
exit(1);
    }

if (stat(argv[1], &statbuf) == -1) {
perror("stat");
exit(1);
    }
```

```
if ((fd = open(argv[1], O_RDWR)) == -1) {
perror("open");
exit(1);
    }

addr = mmap(NULL, statbuf.st_size, PROT_READ|PROT_WRITE,
             MAP_SHARED, fd, (off_t)0);
if (addr == MAP_FAILED) {
perror("mmap");
exit(1);
    }
close(fd);

switch (pid = fork()) {
case -1 :  /* fork failed */
       perror("fork");
       exit(1);
       break;
case 0 :   /* child process */
       printf("1. Child Process :addr=%s", addr);
       sleep(1);
       addr[0] = 't';
       printf("2. Child Process :addr=%s", addr);
       sleep(2);
       printf("3. Child Process :addr=%s", addr);
       break;
default :   /* parent process */
       printf("1. Parent process :addr=%s", addr);
       sleep(2);
       printf("2. Parent process :addr=%s", addr);
       addr[1] = 'H';
       printf("3. Parent process :addr=%s", addr);
       break;
}

return 0;
}
```

```
$ cat test.txt
This is test.
```

와 같은 test.txt파일로

```
$ ./6-6 test.txt
```

하여 실행 결과를 적고, 설명하시오.

7. (6-7.c, 6-7) 위의 6-6.c를 바탕으로 "test.txt"라는 파일을 위한 메모리 맵 영역을 할당하고 child 프로세스에서 해당 영역에 "Child test string"을 써 넣고 parent 프로세스는 해당 영역을 읽어 대문자는 소문자로 소문자는 대문자로 바꾸어 stdout으로 출력하는 프로그램 6-7.c를 작성하시오. 이 프로그램을 컴파일하여 수행하고 결과를 보이시오.

끝.