2016025469 컴퓨터공학과 서건식

운영 체제 HW#2

제출 일자 : 2020/04/06

A. 과제 A

1. 부모 프로세스, 자식 프로세스1, 자식 프로세스2 코드 설명

1) 부모프로세스

```c
#include <stdio.h>
#include <string.h>
#include <unistd.h>
#include <sys/types.h>
#include <sys/wait.h>
#include <stdlib.h>

int main(void)
{
        pid_t pid1;
        pid_t pid2;
        int status1;
        int status2;
        pid1 = fork();//Child Process is made.
        if (pid1 < 0)

        {
                perror("fork error : ");
                exit(0);
        }
        else if (pid1 == 0)
        {
                execl("hw2_child2",NULL);
        }

        else
        {
                pid2 = fork();
                if(pid2 <0)
                {
                        perror("fork error : ");
                        exit(0);
                }
                else if (pid2 == 0)
                {
                        execl("hw2_child1",NULL);
                }
                else{
                        printf("Waiting for child processes\n");
                        pid1 = waitpid(pid1,&status1,0);
                        pid2 = waitpid(pid2,&status2,0);
                        if(0 == (status1&0xff)&& 0 == (status2&0xff)){
                                printf("Child processes are exit.\n");
                                exit(0);
                        }
                        else
```

```
                {
                        exit(2);
                }


        }
    }
    return 0;
```

57,0-1

2개의 자식 프로세스를 생성합니다.

조건문으로 경우를 나눴으며, 첫번째 자식은 hw2_child2를 실행합니다.

else문으로 들어가면 부모 프로세스인 경우이며, 부모 프로세스인 경우 자식을 한번 더 생성합니다. 거기서 또 조건문으로 경우를 나눠주고, 여기서 0인 경우 hw2_child1을 실행합니다. Else의 경우 부모 프로세스이므로 waitpid로 status값을 받아 process들이 모두 끝날 때 까지 기다린 후에 조건문에 걸리면 exit(0)코드로 부모 프로세스를 종료합니다.

2) Hw2_child1.c

```c
#include <stdio.h>
#include <string.h>
#include <unistd.h>
#include <sys/types.h>
#include <sys/wait.h>
#include <stdlib.h>



int main(){
        int i;
        FILE* file;
        file = fopen("text.txt","w");
        for(i=0 ; i<18; i++){
                char input[5];
                sprintf(input,"%d",i);
                fseek(file,0,0);
                fprintf(file,"%s\n",input);
                printf("Child1 wrote %d\n",i);
                fflush(file);
        }
        fclose(file);
}
```

과제의 조건은 다음과 같습니다.

자식 프로세스 1 test.txt 파일을 w 모드로 열어 1 초 간격으로 숫자와 개행문자를 쓰며 숫자는 1 씩 증가

자식 프로세스 2 test.txt 파일을 r 모드로 열어 2 초 간격으로 첫번째 줄을 읽고 이를 화면에 출력

자식 프로세스2(hw2_child2.c)가 첫번째 줄을 읽어야 하므로 hw2_child1.c는 매번 첫번째 줄에 입력값을 덮어씌웁니다. Fseek 함수로 SEEK_SET(파일 시작점)으로 포인터를 이동하여 덮어씌웁니다.

조건문의 시작 시점에서 1초간 기다린 후 씁니다.

또한, fflush로 버퍼에 있는 값은 조건문이 끝나는 시점에 (한 숫자씩) 모두 디스크에 반영합니다.

3) Hw2_child2.c

```c
#include <stdio.h>
#include <string.h>
#include <unistd.h>
#include <sys/types.h>
#include <sys/wait.h>
#include <stdlib.h>


int main(){
        int i;
        for(i=0 ; i<18 ; i+=2){
                sleep(2);
                FILE* file;
                file = fopen("text.txt","r");
                char output[40];
                fgets(output,10,file);
                printf("%s",output);
                fclose(file);
        }

}
```

fgets를 사용해 파일을 읽어오는 프로그램입니다.

조건문은 2씩 증가합니다. 이유는 2초동안 hw2_child1에서는 i가 2씩 증가하기 때문에 그 속도를 맞춰주기 위함입니다.

또한 file 구조체의 선언과 fopen, fclose를 조건문 내에 삽입합니다.

그 이유는 fflush를 사용하긴 했지만 좀 더 확실히 최신으로 동기화된 파일을 가져오기 위함입니다.

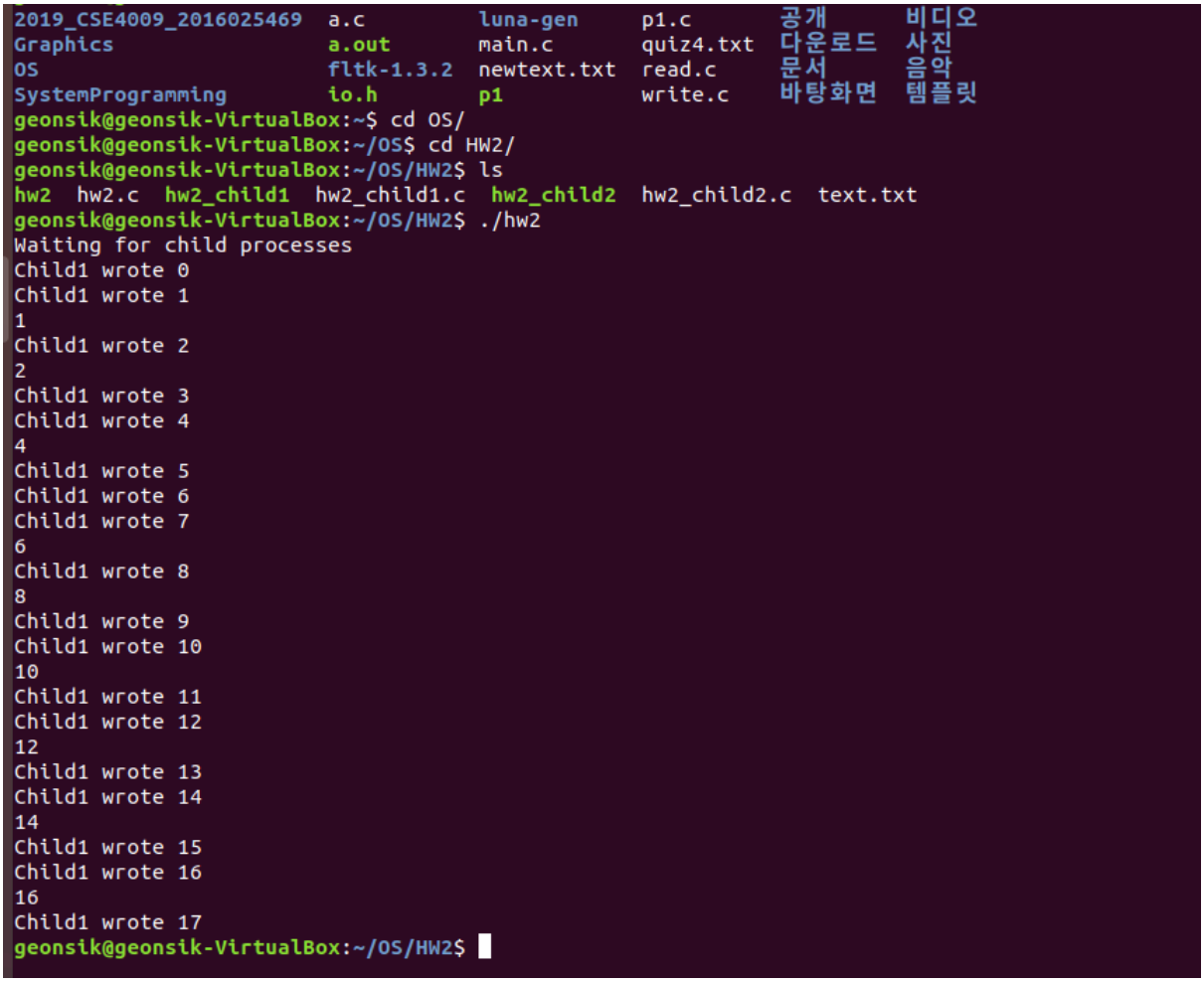
2. fork-exec 구조 설명

fork-exec의 구조는 1번 부모프로세스 코드 설명에서 했습니다.

3. 컴파일 방법 설명

Gcc를 사용하였습니다.

4. 실행 결과



B. 과제 B

1. IPC 내용

IPC = Inter Process Communication -> 프로세스 간 통신

프로세스는 기본적으로 완전히 독립된 실행 객체이다. 서로간의 통신을 위

해서 운영체제의 커널영역에서는 IPC라는 내부 프로세스간 통신을 제공한다.

1) **PIPE**

   현실세계의 파이프처럼 작동함. 한쪽은 읽기, 한쪽은 쓰기만 가능항여 반이중 통신이라고 불림.

2) **Named PIPE**

   PIPE와 다르게 부모 프로세스와 무관하게 전혀 다른 프로세스와 통신이 가능함.

3) **Message Queue**

   선입선출의 구조이며, 메모리 상에 저장되어 있어 언제든 어디서나 꺼낼 수 있다.

4) **Shared Memory**

   데이터 자체를 공유한다.(프로세스 간 메모리 영역을 공유.) -> 가장 빠르다.

5) **Memory Map**

   Shared Memory와 비슷하지만 열린 파일을 메모리에 맵핑시켜서 공유한다는 점에서 다르다.

6) **Socket**

   네트워크를 통해서 다른 프로세스와 소켓 연결을 통해 프로세스간 통신이 가능하다.

7) **Semaphore**

   프로세스 간 데이터를 동기화 하고 보호하는데 목적을 둔 IPC