운영체제

4주차 과제

20185109 길홍배

(1) 기본 과제

- fork 함수를 이용해서 부모와 자식 프로세스를 생성한다.

- 자식 프로세스는 exec 함수를 사용해서 현재 프로세스 상태를 출력("ps -al")한다.

1. 코드 작성

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

int main()
{
    pid_t pid;

    pid = fork();

    if(pid < 0) {
        fprintf(stderr, "Fork Failed");
        return 1;
    }

    else if(pid == 0) { execlp("/bin/ps", "ps", "-al", NULL); }

    else
    {
        wait(NULL);
        printf("Child Complete");
    }

    return 0;
}%
(base) chr1s0@gilhongbaeui-MacBookAir week4 %
```

2. 컴파일 및 실행 과정

```
(base) chr1s0@gilhongbaeui-MacBookAir week4 % gcc -o task2 task2.c
(base) chr1s0@gilhongbaeui-MacBookAir week4 % ./task2
  UID   PID  PPID        F CPU PRI NI       SZ    RSS WCHAN     S             ADDR TTY           TIME CMD
  501  1418  1165     4006   0  31  0 408938864   3760 -       Ss+               0 ttys000    0:00.06 /bin/zsh -l
    0  1844  1843     4106   0  31  0 408647952   6144 -       Ss                0 ttys001    0:00.01 login -pf chr1s0
  501  1845  1844     4006   0  31  0 408956640   5024 -       S                 0 ttys001    0:00.12 -zsh
  501  2160  1845     4006   0  31  0 408497216   1392 -       S+                0 ttys001    0:00.00 ./task2
    0  2161  2160     4106   0  31  0 408637552   2208 -       R+                0 ttys001    0:00.00 ps -al
Child Complete%
(base) chr1s0@gilhongbaeui-MacBookAir week4 %
```

(2) 도전 과제

- fork 함수를 사용해서 프로세스를 4개 생성한다.
- 부모 프로세스는 자식 프로세스의 pid를 배열("int pid_array[5]")에 보관한다.
- 자식 프로세스는 생성후에 sleep()함수를 이용하여 5초에서 20초 사이에서 랜덤하게 sleep한다.
- 잠에서 깨어난 후에 exec 함수를 사용해서 ps, ls, df, cal를 각각 수행시킨다.
- 자식 프로세스가 종료하면 부모 프로세스는 배열에 저장된 자식 프로세스의 아이디를 출력시키고 종료한다.

1. 코드 작성 및 코드 설명

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

#define CHILD_PROCESS_COUNT 4 // 최대 자식 프로세스 수 정의

int get_random(); // 함수 선언

enum CommandList /* 명령어 실행을 위한 열거형 */
{
    PS = 0,
    LS,
    DF,
    CAL
}cl;
int main()
 {
    /*
        pid 배열을 선언해 자식 프로세스 저장
        pid_array 배열을 선언해 자식 프로세스의 pid 저장
    */
    pid_t pid[CHILD_PROCESS_COUNT];
    int pid_array[CHILD_PROCESS_COUNT];

    int i = 0;
    int sleep_time;

    for (i = 0; i < CHILD_PROCESS_COUNT; i++)
    {
        pid[i] = fork();

        if(pid[i] < 0)
        {
            perror("Fork Failed");
            return 1;
        }
```

```c
int get_random() /* 랜덤한 sleep시간을 생성하는 함수 */
{
    int random;

    srand(time(NULL));
    random = (rand() % 15) + 5;

    return random;
}
```

```c
        /*
            자식 프로세스임이 확인 된 경우 pid를 배열에 저장
            이후 5 ~ 20초 사이의 sleep을 진행,
            이후 자식 프로세스의 순서에 따라 명령어 실행
        */

        else if(pid[i] == 0)
        {
            sleep_time = get_random();
            pid_array[i] = getpid();

            printf("Child forked, Sleeping for %d second...\n", sleep_time);
            sleep(sleep_time);
            if(i == PS)
                execlp("/bin/ps", "ps", NULL);
            else if(i == LS)
                execlp("/bin/ls", "ls", NULL);
            else if(i == DF)
                execlp("/bin/df", "df", NULL);
            else if(i == CAL)
                execlp("cal", "cal", NULL);
        }
```

```c
            /*
                자식 프로세스 종료 이후
                pid_array에 저장된 pid 출력
            */
            else
            {
                wait(NULL);
                printf("Child Complete pid: %d\n\n", pid_array[i]);
            }
        }

        return 0;
    }
```

fork() 함수를 이용하여 프로세스 생성 이후 배열에 저장, 생성된 프로세스가 자식 프로세스인지 확인, 자식프로세스임이 확인 된 경우 getpid()를 이용해 pid_array 배열에 저장, get_random() 함수를 이용해 5 ~ 20 초 사이의 값을 받아와 sleep() 함수 실행, 열거형에 저장된 값을 이용해 ps, ls, df, cal 차례대로 실행, 이후 프로세스가 종료 된 뒤 pid_array 에 있는 id 값 출력

(2) 실행 결과

```
Child forked, Sleeping for 7 second...
  PID TTY           TIME CMD
 1418 ttys000    0:00.06 /bin/zsh -l
 1845 ttys001    0:00.15 -zsh
 2395 ttys001    0:00.00 ./task1
Child Complete pid: 573536

Child forked, Sleeping for 11 second...
4주 차 _운 영 체 제 _20185109_길 홍 배 .docx
task1
task1.c
task1.dSYM
task2
task2.c
task2.dSYM
~$차 _운 영 체 제 _20185109_길 홍 배 .docx
Child Complete pid: 1

Child forked, Sleeping for 13 second...
Filesystem     512-blocks      Used Available Capacity iused      ifree %iused  Mounted on
/dev/disk3s1s1  478724992  29660712 204052544    13%  500632 1020262720    0%   /
devfs                 402       402         0   100%     696          0  100%   /dev
/dev/disk3s6    478724992        40 204052544     1%       0 1020262720    0%   /System/Volumes/VM
/dev/disk3s2    478724992    917664 204052544     1%    1474 1020262720    0%   /System/Volumes/Preboot
/dev/disk3s4    478724992     13832 204052544     1%      47 1020262720    0%   /System/Volumes/Update
/dev/disk1s2      1024000     12328    986088     2%       3    4930440    0%   /System/Volumes/xarts
/dev/disk1s1      1024000     15096    986088     2%      32    4930440    0%   /System/Volumes/iSCPreboot
/dev/disk1s3      1024000       768    986088     1%      37    4930440    0%   /System/Volumes/Hardware
/dev/disk3s5    478724992 242253176 204052544    55% 1237274 1020262720    0%   /System/Volumes/Data
map auto_home           0         0         0   100%       0          0  100%   /System/Volumes/Data/home
Child Complete pid: 1876440576

Child forked, Sleeping for 14 second...
      3월  2022
일  월  화  수  목  금  토
       1   2   3   4   5
 6   7   8   9  10  11  12
13  14  15  16  17  18  19
20  21  22  23  24  25  26
27  28  29  30  31

Child Complete pid: 1
```