오픈소스 소프트웨어 실습

Open-Source Software Lab

#6

실습 담당 조교 연락처

◆실습 조교 : 정만성

◆연구실 : 학연산클러스터 604호

◆과제 제출 및 문의 - 이메일

◆이메일 : wjdakstjd@hanyang.ac.kr

Process의 개념

- 수행 중인 프로그램 (Program in execution)
- 각 프로세스는 고유의 프로세스 번호 PID를 가짐
- 각 프로세스는 text(code), data, stack 영역을 메모리에 할당 받음
- 프로세스(child)는 다른 프로세스(parent)에 의해 생성됨
- 두 프로세스는 프로세스간 통신 (IPC) 기능을 이용하여 정보를 주고 받을 수 있음 ex) signal, pipe, socket, …

프로세스 상태 보기: ps

$ ps [-옵션]

현재 시스템 내에 존재하는 프로세스들의 실행 상태를 요약해서 출력한다.

```
~/osw$ ps
    PID TTY          TIME CMD
     15 pts/1    00:00:00 bash
     50 pts/1    00:00:00 ps
~/osw$ ps -u
USER         PID %CPU %MEM    VSZ   RSS TTY      STAT START   TIME COMMAND
runner        12  0.0  0.0  20184  3232 pts/0    Ss+  21:11   0:00 bash --norc
runner        15  0.0  0.0  20184  3788 pts/1    Ss   21:11   0:00 bash --norc
runner        51  0.0  0.0  36080  3244 pts/1    R+   21:11   0:00 ps -u
```

프로세스 출력정보

항목	의미
UID	프로세스를 실행시킨 사용자 ID
PID	프로세스 번호
PPID	부모 프로세스 번호
C	프로세스의 우선순의
STIME	프로세스의 시작 시간
TTY	명령어가 시작된 터미널
TIME	프로세스에 사용된 CPU 시간
CMD	실행되고 있는 명령어(프로그램) 이름

프로세스 관계 확인 : pstree

```
~/osw$ pstree
init─┬─bash───pstree
     ├─bash
     ├─querydb───10*[{querydb}]
     ├─sleep
     └─12*[{init}]
~/osw$
```

특정 프로세스 리스트 : pgrep

```
~/osw$ pgrep bash
12
15
~/osw$ pgrep -l bash
12 bash
15 bash
```

쉘 재우기

$ sleep 초

명시된 시간만큼 프로세스 실행을 중지시킨다.

```
~/osw$ (echo start; sleep 5; echo end;)
start
end
~/osw$
```

강제 종료

강제 종료 : ctrl – c

실행 중지 : ctrl - z

```
~/osw$ (echo start; sleep 5; echo end;)
start
^C
~/osw$ (echo start; sleep 5; echo end;)
start
^Z
[1]+  Stopped                 ( echo start; sleep 5; echo end )
~/osw$
```

전면 작업의 후면 전환 : bg

$ bg %작업번호

작업번호에 해당하는 중지된 작업을 후면 작업으로 전환하여 실행한다.

```
~/osw$ (sleep 20; echo DONE)
^Z
[1]+  Stopped                 ( sleep 20; echo DONE )
~/osw$ bg %1
[1]+ ( sleep 20; echo DONE ) &
~/osw$ jobs
[1]+  Running                 ( sleep 20; echo DONE ) &
~/osw$ DONE
```

프로세스 끝내기 : kill

$ kill 프로세스 번호

$ kill %작업번호

프로세스 번호(혹은 작업 번호)에 해당하는 프로세스를 강제로 종료 시킨다.

```
~/osw$ (sleep 10; echo DONE)
^Z
[1]+  Stopped                 ( sleep 10; echo DONE )
~/osw$ kill %1
~/osw$ jobs
[1]+  Terminated              ( sleep 10; echo DONE )
~/osw$
```

Process 관련 시스템 호출

- fork : 자신의 프로세스를 복제하여 child 프로세스를 생성
- exec : 자신의 프로세스에 다른 프로그램을 덮어 씀
- wait : child 프로세스가 종료할 때까지 기다림
- exit : 자신의 프로세스를 종료하며, 상태 정보를 반환

System Call – fork()

Synopsis

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

pid_t fork(void);
```

- 자식 프로세스를 생성함
- Return value
  - > parent process : child proces의 PID
  - > child process : 0
  - > 실패 : -1

fork() example

```c
#include <stdio.h>
#include <unistd.h>
#include <stdlib.h>

int main(int argc, char **argv) {

  int pid;
  if ((pid = fork()) > 0) {
    // getpid() 현재 프로세스의 프로세스ID를 되돌려 줌
    printf("부모 프로세스 %d : %d\n", getpid(), pid);
    return 0;
  } else if (pid == 0) {
    printf("자식 프로세스 %d\n", getpid());
    return 0;
  } else {
    perror("fork error : ");
    exit(0);
  }
  return 0;
}
```

System Call – exec()

Synopsis

#include <unistd.h>

Int execl(const char *path, const char *arg0, ..., const char *argN, (char *)0);

Int execlp(const char *file, const char *arg0, ..., const char *argN, (char *)0);

Int execv(const char *path, char *const argv[]);

Int execvp(const char *file, char *const argv[]);

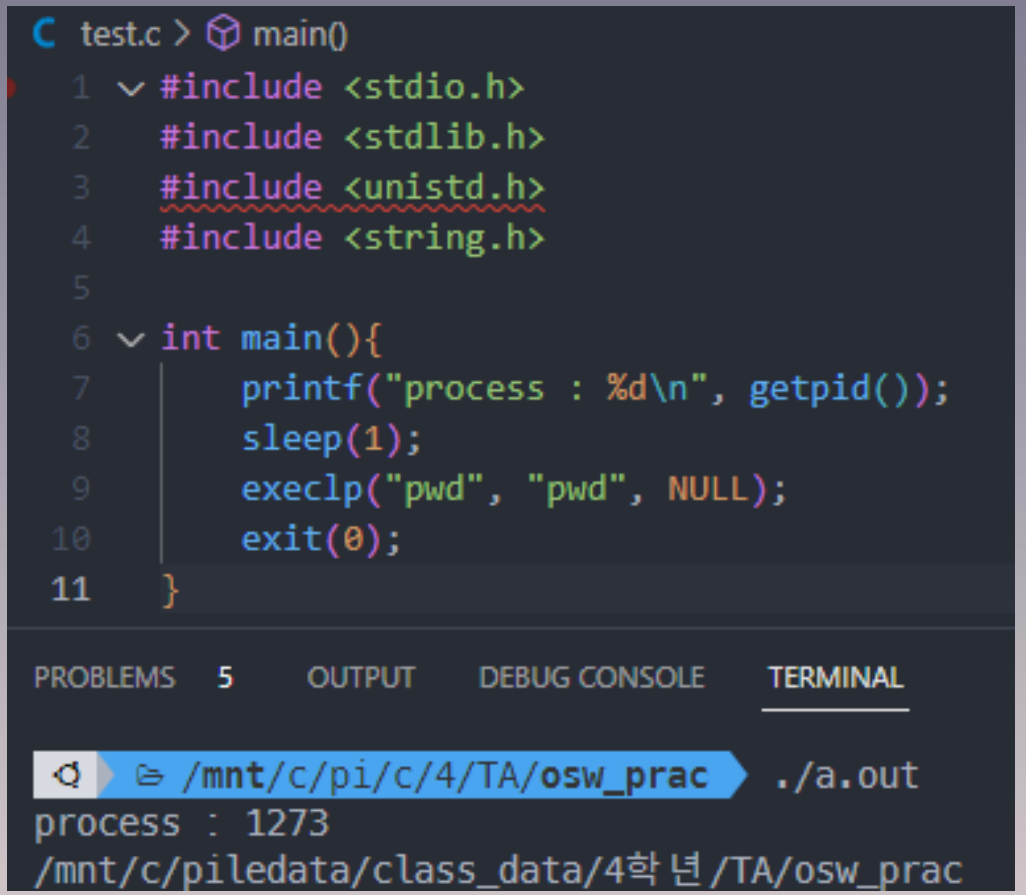
...

- 현재 프로세스 이미지를 새로운 프로세스 이미지로 바꿈
  - > 새 프로그램의 수행이 시작되어, 새 data와 stack 형성
  - > 이전에 open된 fd는 exec 이후에도 사용 가능
  - > 이미 open된 fd를 close할 수도 있음

exec() example

```c
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <string.h>

int main() {
  printf("원래 프로세스: %d\n", getpid());
  sleep(1);
  execl("/bin/sh", "sh", NULL);
  exit(0);
}
```



exec() family

exec() family
int execl(const char *pathname, const char *arg, .../* (char *) NULL */);
int execv(const char *pathname, char *const argv[]);
int execle(const char *pathname, const char *arg, .../*, (char *) NULL, char *const envp[] */);
int execve(const har* pathname, char *const argv[], char *const ecnp[]);
int execlp(const char *file, const char *arg, .../* (char *) NULL */);
int execvp(const char *file, char *const argv[]);
int execvpe(const char *file, char *const argv[],char *const envp[]);

알파벳	설명
l	인자를 배열로 받음
v	인자를 vector(char *argv[])형태로 받음(NULL 불필요)
p	기본 환경 변수의 경로를 이용함
e	environment를 입력받음

System Call – wait()

Synopsis

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

pid_t wait(int *status);
```

- 자식 프로세스가 종료될 때까지 해당 영역에서 부모 프로세스가 sleep 모드로 기다림

fork() & exec() 실습

```c
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <fcntl.h>
#include <sys/wait.h>
#include <sys/types.h>
#define PERM 0644

int main(int argc, char *argv[]){
    pid_t pid;
    int fd1;
    if(argc < 3){
        printf("arg error. Please insert the existing file and the target file.\n");
    ① execlp(                                    );
        printf("If this line is printed, there is a problem with execlp().\n");
    }

  ② if                                    
        fd1 = open(argv[2], O_CREAT|O_WRONLY|O_TRUNC, PERM);
        close(fd1);
    ③ execlp(                                                  );
        printf("If this line is printed, there is a problem with execlp().\n");
    }
    wait(NULL);

  ④ if                                    
        printf("--------------------------------\n");
        printf("contents of %s\n", argv[1]);
        printf("--------------------------------\n");
    ⑤ execlp(                                        );
        printf("If this line is printed, there is a problem with execlp().\n");
    }

    wait(NULL);
    printf("--------------------------------\n");
    printf("contents of %s\n", argv[2]);
    printf("--------------------------------\n");
  ⑥ execlp(                                    );
    printf("If this line is printed, there is a problem with execlp().\n");
    return 0;
}
```

요구사항

1. 전달 된 인자가 부족할경우, “ls –al” 명령어가 실행되어야 함.
2. fork()를 이용하여 자식 프로세스에게 “cp 인자1 인자2” 명령어를 실행시켜야 함.
3. cp 명령어 이후 다시 fork()를 이용하여 인자1의 내용을 앞부분만 출력한다
4. 부모 프로세스는 최종적으로 인자2의 내용의 앞부분을 출력한다.

fork() & exec() 실습

```
oswTA@sseojinn:~$ vim week6.c
oswTA@sseojinn:~$ gcc week6.c
oswTA@sseojinn:~$ ./a.out hlist.txt target.txt
--------------------------------
contents of hlist.txt
--------------------------------
    1  history
    2  ls
    3  ./a.out
    4  history
    5  exit
    6  su
    7  ls
    8  cd ..
    9  ls
   10  pwd
--------------------------------
contents of target.txt
--------------------------------
    1  history
    2  ls
    3  ./a.out
    4  history
    5  exit
    6  su
    7  ls
    8  cd ..
    9  ls
   10  pwd
```

```
oswTA@sseojinn:~$ ./a.out
arg error. Please insert the existing file and the target file.
total 128
drwxr-xr-x  4 oswTA oswTA 4096  4월   9 22:55 .
drwxr-xr-x 75 root  root  4096  4월   5 12:38 ..
-rwxrwxr-x  1 oswTA oswTA 8560  4월   9 22:55 a.out
-rw-------  1 oswTA oswTA 4227  4월   5 12:40 .bash_history
-rw-r--r--  1 oswTA oswTA  220  4월   5  2018 .bash_logout
-rw-r--r--  1 oswTA oswTA 3771  4월   5  2018 .bashrc
drwx------  2 oswTA oswTA 4096  3월  18 20:28 .cache
-rw-rw-r--  1 oswTA oswTA  540  3월  26 21:35 cpfile.c
-rw-r--r--  1 oswTA oswTA 8980  4월  16  2018 examples.desktop
drwx------  3 oswTA oswTA 4096  3월  18 20:28 .gnupg
-rw-r--r--  1 oswTA oswTA  985  3월  26 23:49 hlist2.txt
-rw-rw-r--  1 oswTA oswTA  985  3월  26 23:49 hlist.txt
-rw-rw-r--  1 oswTA oswTA 4183  3월  27 01:51 input.txt
-rw-r--r--  1 oswTA oswTA  807  4월   5  2018 .profile
-rwxrwxr-x  1 oswTA oswTA 8568  3월  26 23:49 simplecp
-rw-r--r--  1 oswTA oswTA  985  4월   9 22:55 target.txt
-rwxr-xr-x  1 oswTA oswTA  568  3월  29 10:27 test2.c
-rwxrwxr-x  1 oswTA oswTA  568  3월  28 10:22 test.c
-rw-------  1 oswTA oswTA 9146  4월   9 22:54 .viminfo
-rw-rw-r--  1 oswTA oswTA 1247  4월   5 10:34 week5.c
-rw-r--r--  1 oswTA oswTA 1189  4월   5 10:38 week5test.c
-rw-rw-r--  1 oswTA oswTA 1312  4월   9 22:54 week6.c
```

끝