오픈소스 소프트웨어 실습

Open-Source Software Lab

#7

실습 담당 조교 연락처

◆실습 조교 : 김민곤

◆연구실 : 학연산클러스터 601호

◆이메일 : phenix235@hanyang.ac.kr

메일양식: [오픈소스]학번_수업일(20190000)_이름

프로세스 크기: size

$ size [실행파일]

실행파일의 각 영역의 크기를 알려준다.

실행파일을 지정하지 않으면 a.out을 대상으로 한다.

```
[1111222333@node1 ~]$ size /bin/ls
   text    data     bss     dec     hex filename
 103359    4792    3360  111511   1b397 /bin/ls
```

프로세스 트리: pstree

$ pstree

실행중인 프로세스들의 부모, 자식관계를 트리 형태로 출력한다.

```
[1111222333@node1 ~]$ pstree
systemd─┬─ModemManager───2*[{ModemManager}]
        ├─5*[a]
        ├─24*[a.out]
        ├─abrt-dbus───3*[{abrt-dbus}]
        ├─2*[abrt-watch-log]
        ├─abrtd
        ├─accounts-daemon───2*[{accounts-daemon}]
        ├─10*[al]
        ├─20*[alarm]
        ├─21*[alarm.out]
        ├─alsactl
        ├─at-spi-bus-laun─┬─dbus-daemon───{dbus-daemon}
        │                 └─3*[{at-spi-bus-laun}]
        ├─at-spi2-registr───2*[{at-spi2-registr}]
        ├─atd
```

fork

사용법

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

pid_t fork(void); pid_t vfork(void);
```

자식 process를 생성

Return value

[parent process : child process #]

[child process : 0]

fork: 부모와 자식이 다른 메모리 공간 사용

vfork: 부모와 자식이 같은 메모리 공간 공유

실패경우 1 - 시스템에서 허용하는 프로세스 개수를 초과한 경우

2 – 개별 사용자가 동시에 수행할 수 있는 프로세스 초과

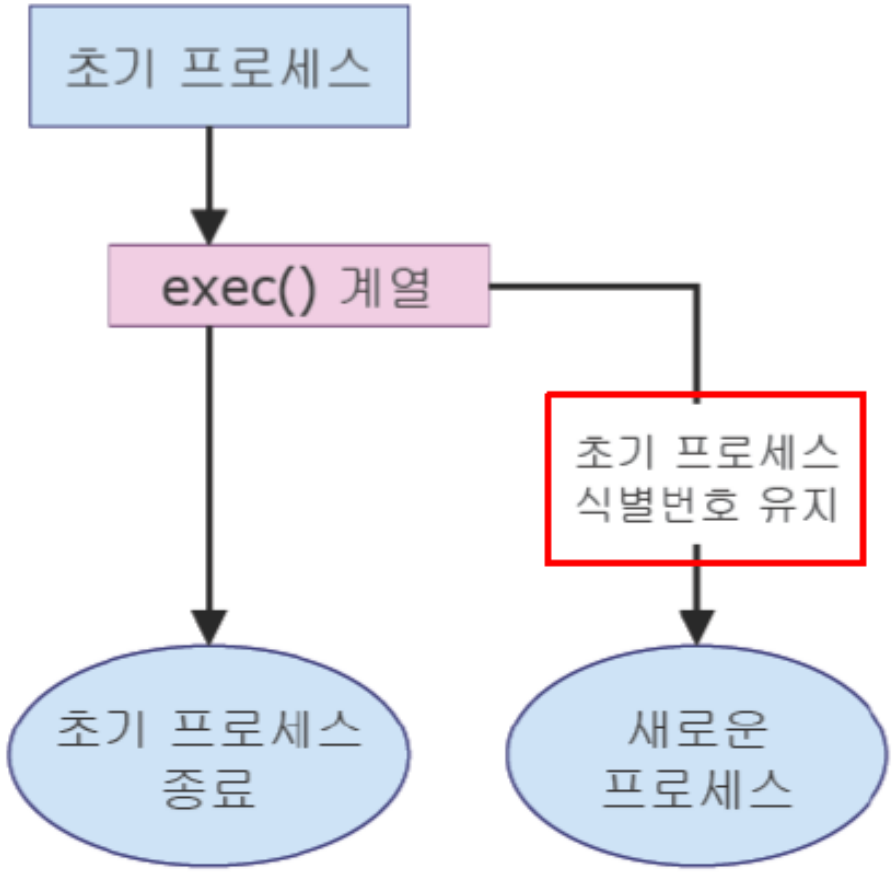
fork

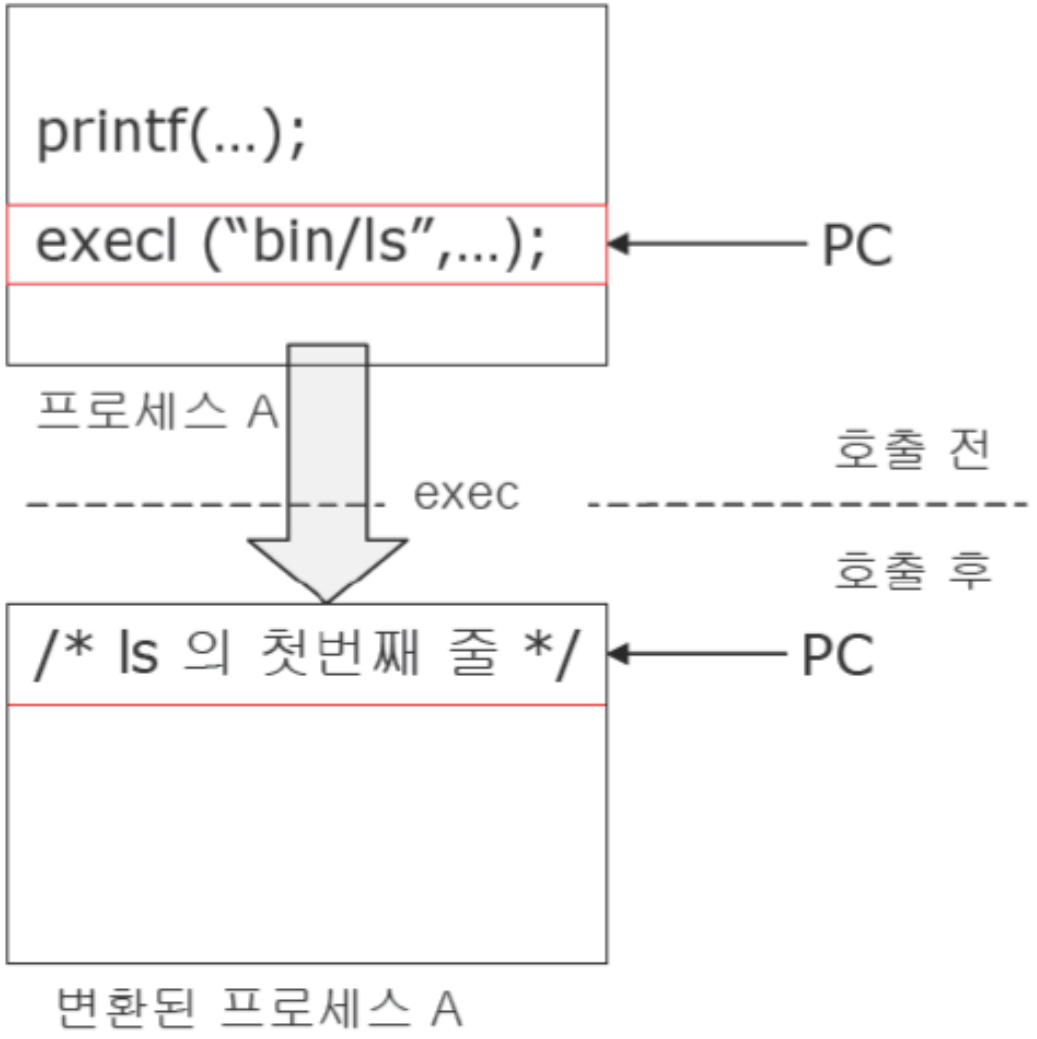
```c
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>

int main(int argc, char* argv[])
{
    pid_t pid = fork();
    if (pid > 0){
                printf("parent process %d : %d\n", getpid(), pid);
    } else if (pid == 0){
        printf("child process %d\n", getpid());
    } else if (pid < 0){
        perror("fork error");
        exit(0);
    }
}
```

```
[1111222333@node1 class7]$ ./fork
child process 13299
parent process 13298 : 13299
[1111222333@node1 class7]$ ./fork
parent process 13356 : 13357
child process 13357
```

exec


초기 프로세스
exec() 계열
초기 프로세스
식별번호 유지
초기 프로세스
종료
새로운
프로세스


printf(...);
execl ("bin/ls",...);
PC
프로세스 A
호출 전
exec
호출 후
/* ls 의 첫번째 줄 */
PC
변환된 프로세스 A

exec

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

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

```
[1111222333@node1 class7]$ ./exec
original process: 14088
sh-4.2$ echo $$
14088
sh-4.2$ exit
exit
```

open

사용법

```
#include <sys/types.h>
#include <sys/stat.h>
#include <fcntl.h>

int open (const char *pathname, int flag, mode_t mode);
```

Return value – [성공시 : new file descriptor] [실패시 :- 1]

Flag <fcntl.h>

O_RDONLY	읽기 전용
O_WRONLY	쓰기 전용
O_RDWR	읽기 쓰기
O_CREAT	파일이 존재 하지 않으면 생성
O_EXCL	O_CREAT와 함께 사용되며 파일이 존재 시 에러처리
O_TRUNC	파일이 존재 시 잘라버림
O_APPEND	파일의 뒷부분에 추가

creat

사용법

```
#include <sys/types.h>
#include <sys/stat.h>
#include <fcntl.h>

int creat(const char *pathname, mode_t mode);
```

: 새 파일 생성시 사용
: open 에서 O_CREAT|O_WRONLY|O_TRUNC flag와 같음

close

사용법

```
#include <sys/types.h>
#include <sys/stat.h>
#include <fcntl.h>

int close(int fd);
```

: 파일 사용을 끝냈음을 시스템에게 알림

Return value - [성공시 : 0]
[실패시 : -1]

read

사용법

```
#include <unistd.h>

ssize_t read(int fd, void *buf, size_t count);
```

Return value – [성공시 : number of bytes read]
[End of file : 0]
[실패시 :- 1]

: 파일로 부터 임의의 byte를 버퍼로 복사하는데 사용

write

사용법

```
#include <unistd.h>

ssize_t write(int fd, const void *buf, size_t count);
```

Return value – [성공시 : number of bytes read]
[End of file : 0]
[실패시 :- 1]

: 버퍼로부터 임의의 byte를 파일에 쓰는데 사용

Simpleio.c

1. test.in 파일 생성
2. 아무내용이나 입력
3. 컴파일 후 실행
4. test.out 파일 생성 확인
5. test.out 파일 내용 확인

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

#define BSIZE 1024
#define FPERM 0644

int main()
{
    int fd1, fd2, n;
    char buf[BSIZE];
    if((fd1=open("test.in", O_RDONLY)) < 0){
        perror("file open error");
        exit(1);
    }
    if((fd2=creat("test.out", FPERM)) < 0){
        perror("file creation error");
        exit(1);
    }
    while((n = read(fd1, buf, BSIZE)) > 0)
        write(fd2, buf, n);
    close(fd1);
    close(fd2);
}
```

실습 과제

- ✓ 연습코드인 simpleio.c의 내용을 기반으로 한 파일의 내용을 다른 파일로 복사하는 simplecp.c를 구현
- ✓ 원본 파일을 RDONLY로 열기
- ✓ 대상 파일을 생성 (creat 대신 open 사용)
- ✓ 원본 파일의 끝에 도달할 때까지 파일을 읽어 대상 파일에 기록
- ✓ 두파일 모두 닫음
- ✓ 실행 파일이 simplecp일 경우

  실행 예) $ ./simplecp [source] [destination]
- ✓ Simplecp.c 작성 화면과 실행화면 캡쳐(총2개)해서 양식 맞춰 메일

끝