

12201922

이규민

3. Homework

0) Try ex0 below. Who is the parent of ex0?

ex0.c:

```
#include <stdio.h>
#include <sys/types.h>
#include <unistd.h>
void main(){
    int x,y;
    x=getpid();          // my pid
    y=getppid();         // parent's pid
    printf("PID:%d PPID:%d\n", x, y);
    for(;;);  // to make this process alive
}
```

```
$ gcc -o ex0 ex0.c
```

Run ex0 with &. "&" puts process in the background so that you can issue next command to the shell.

```
$ ex0&
```

.....

Now confirm with "ps -f".

```
$ ps -f
```

.....

Now kill ex0.

```
$ kill xxxx
```

,where xxxx is the pid of ex0.

```
kyumin@DESKTOP-NUDFAPK ~  
$ ps -f  
    UID      PID     PPID  TTY          STIME COMMAND  
kyumin    1434     1433  pty0    09:06:03 /usr/bin/bash  
kyumin    1457     1434  pty0    09:17:35 /usr/bin/ps  
kyumin    1433         1  ?       09:06:02 /usr/bin/mintty  
  
kyumin@DESKTOP-NUDFAPK ~  
$ ex0 &  
[1] 1458  
  
kyumin@DESKTOP-NUDFAPK ~  
$ PID:1458 PPID:1434  
ps -f  
    UID      PID     PPID  TTY          STIME COMMAND  
kyumin    1459     1434  pty0    09:17:57 /usr/bin/ps  
kyumin    1434     1433  pty0    09:06:03 /usr/bin/bash  
kyumin    1458     1434  pty0    09:17:48 /cygdrive/c/Users/kyumin/AppData/Roam  
ing/SPB_Data/ex0  
kyumin    1433         1  ?       09:06:02 /usr/bin/mintty
```

Ex0 프로그램을 실행하면 pid는 1458, ppid는 1434라고 출력된다. Ps -f 명령어로 확인해 보면 동일한 pid임을 확인할 수 있다.

1) Try ex1 below. Why do we have two hello's? What are the PID of ex1 and ex1's child? Who is the parent of ex1?

ex1.c:

```
#include <stdio.h>  
#include <sys/types.h>  
#include <unistd.h>  
void main(){  
    int x;  
    x=fork();  
    printf("hello\n");  
    for(;;);  
}
```

```
$ gcc -o ex1 ex1.c
```

```
$ ex1&
```

```
hello
```

```
hello
```

```
$ ps -f
```

```
.....
```

```
$ kill xxxx(pid of ex1) yyyy(pid of ex1's child)
```

```

kyumin@DESKTOP-NUDFAPK ~
$ ex1 &
[1] 1797

kyumin@DESKTOP-NUDFAPK ~
$ hello
hello
ps -f

```

UID	PID	PPID	TTY	STIME	COMMAND
kyumin	1784	1783	pty0	09:22:06	/usr/bin/bash
kyumin	1783	1	?	09:22:05	/usr/bin/mintty
kyumin	1798	1797	pty0	09:24:30	/cygdrive/c/Users/kyumin/AppData/Roam
kyumin	1797	1784	pty0	09:24:30	/cygdrive/c/Users/kyumin/AppData/Roam
kyumin	1799	1784	pty0	09:24:33	/usr/bin/ps

```

kyumin@DESKTOP-NUDFAPK ~
$ kill 1797 1798

kyumin@DESKTOP-NUDFAPK ~
$ ps -f

```

UID	PID	PPID	TTY	STIME	COMMAND
kyumin	1784	1783	pty0	09:22:06	/usr/bin/bash
kyumin	1783	1	?	09:22:05	/usr/bin/mintty
kyumin	1800	1784	pty0	09:26:39	/usr/bin/ps

[1]+ Terminated ex1

Ex1을 실행하면 hello 메시지가 두 번 출력된다. 그 이유는 fork 코드가 실행되면서 프로세스를 복제하게 된다. 그리고 스케줄러가 프로세스를 모두 실행하면서 부모와 자식 모두 printf("hello\n") 코드가 실행되므로 hello가 두 번 출력된다.

Ps -f를 통해 확인해보면 ex1 프로세스가 두 개 생성된 것을 볼 수 있다. 이 중에 pid가 1798인 프로세스를 보면 ppid가 다른 것과 달리 1797이다.

즉 부모의 pid는 1797이고, 자식의 pid는 1798이다.

2) Modify ex1.c such that it prints its own pid and the parent pid. Confirm the result with "ps -f". Who is the parent of ex1? Who is the parent of the parent of ex1? Follow the parent link until you reach PID 1 and show all of them.

```

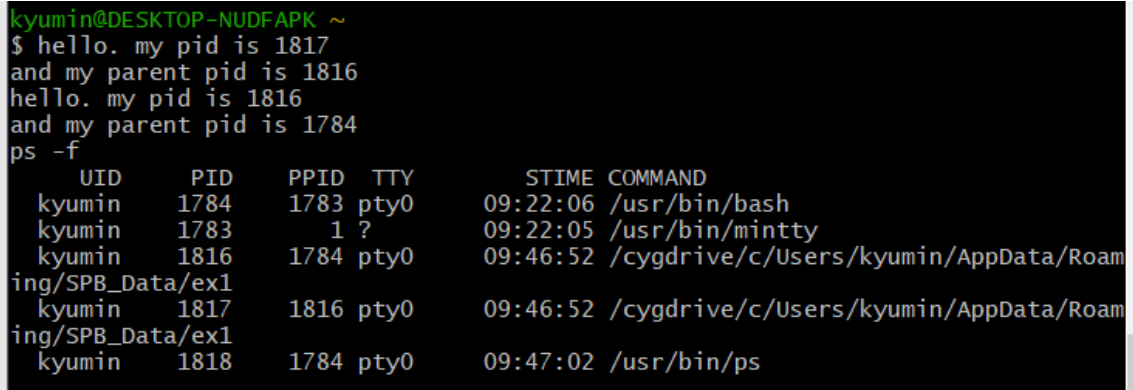
#include <stdio.h>
#include <sys/types.h>
#include <unistd.h>
void main(){
    int x;
    x=fork();

```

```

printf("hello. my pid is %d\n", getpid());
printf("and my parent pid is %d\n", getppid());
for(;;);
}

```



```

kyumin@DESKTOP-NUDFAPK ~
$ hello. my pid is 1817
and my parent pid is 1816
hello. my pid is 1816
and my parent pid is 1784
ps -f

```

UID	PID	PPID	TTY	STIME	COMMAND
kyumin	1784	1783	pty0	09:22:06	/usr/bin/bash
kyumin	1783	1	?	09:22:05	/usr/bin/mintty
kyumin	1816	1784	pty0	09:46:52	/cygdrive/c/Users/kyumin/AppData/Roam
kyumin	1817	1816	pty0	09:46:52	/cygdrive/c/Users/kyumin/AppData/Roam
kyumin	1818	1784	pty0	09:47:02	/usr/bin/ps

출력된 메시지를 보면 1행과 2행에서는 자신의 pid가 1817, 부모의 pid가 1816이라고 출력되었다. Ps -f 명령어를 통해 확인해보면 pid가 1816인 프로세스의 ppid는 1784다.

즉 ex0의 부모 pid는 1816이고, 자식의 pid는 1817이다.

자식의 pid부터 부모의 pid를 계속 따라가보면 1817 -> 1816 -> 1784 -> 1783 -> 1이다.

3) Try below (ex2.c). Which hello is displayed by the parent and which hello is by the child?

```

void main(){
    int x;
    x=fork();
    printf("hello: %d\n", x);
}

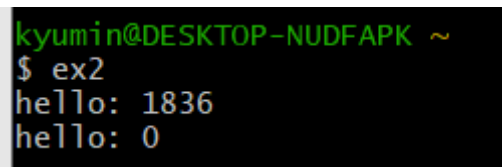
```

```
$ gcc -o ex2 ex2.c
```

```
$ ex2
```

```
hello: 22644
```

```
hello: 0
```



```

kyumin@DESKTOP-NUDFAPK ~
$ ex2
hello: 1836
hello: 0

```

Fork 함수를 사용할 경우 자식은 0을 리턴하고, 부모의 경우 자식의 pid를 리턴한다.

그러므로 x를 출력하는 코드가 실행되었을 때 0이 출력된다면 그것은 자식인 경우다.

위 결과를 보면 “hello: 1836”은 부모임을 알 수 있고, “hello: 0”은 자식임을 알 수 있다.

4) Try below (ex3.c) and show all ancestor processes of ex3 (parent, parent of parent, etc).

```
void main(){
    int x;
    x=fork();
    printf("hello: %d\n", x);
    for(;;) // make the parent and child alive
}
```

```
$ gcc -o ex3 ex3.c
```

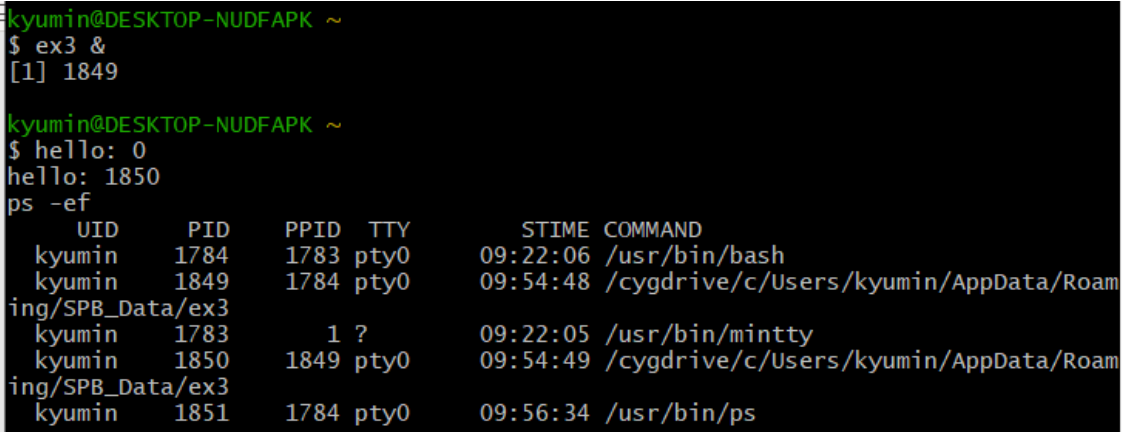
```
$ ex3 &
```

```
hello: 22644
```

```
hello: 0
```

```
$ ps -ef
```

```
.....
```



```
kyumin@DESKTOP-NUDFAPK ~
$ ex3 &
[1] 1849

kyumin@DESKTOP-NUDFAPK ~
$ hello: 0
hello: 1850
ps -ef
  UID      PID    PPID  TTY          STIME   COMMAND
  kyumin   1784    1783  pty0      09:22:06 /usr/bin/bash
  kyumin   1849    1784  pty0      09:54:48 /cygdrive/c/Users/kyumin/AppData/Roam
ing/SPB_Data/ex3
  kyumin   1783      1  ?          09:22:05 /usr/bin/mintty
  kyumin   1850    1849  pty0      09:54:49 /cygdrive/c/Users/kyumin/AppData/Roam
ing/SPB_Data/ex3
  kyumin   1851    1784  pty0      09:56:34 /usr/bin/ps
```

Ex3의 부모의 int x값은 1850이므로 자식의 pid는 1850이다. Ps -ef를 사용해 프로세스를 확인해봤다. 자식부터 pid를 따라가보면 1850 -> 1849 -> 1784 -> 1783 -> 1임을 볼 수 있다.

5) Try below (ex4.c). Which message was displayed by the parent and which one by the child?

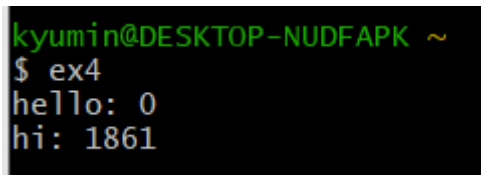
```
void main(){
    int x;
    x=fork();
    if (x==0){
```

```

        printf("hello: %d\n", x);
    }else{
        printf("hi: %d\n", x);
    }
}
$ gcc -o ex4 ex4.c
$ ex4

```

.....



```

kyumin@DESKTOP-NUDFAPK ~
$ ex4
hello: 0
hi: 1861

```

위 코드는 x의 값이 0이라면 hello를 출력하고, 0이 아니면 hi를 출력한다.

자식의 경우 x는 0이므로 hello를 출력할 것이고, 부모의 경우 x는 자식의 pid를 받으므로 hi가 출력될 것이다.

위 사진의 경우에는 윗줄은 자식이 출력한 결과고, 아랫줄이 부모가 출력한 결과다.

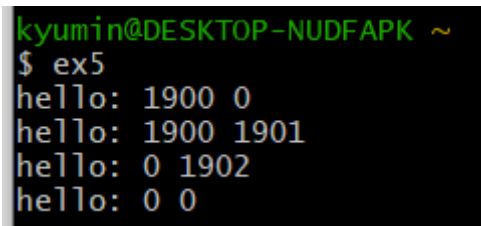
6) Try below (ex5.c). How many hellos do you see? Explain why you have that many hellos. Draw the process tree.

```

void main(){
    int x,y;
    x=fork();
    y=fork();
    printf("hello: %d %d\n", x, y);
}
$ gcc -o ex5 ex5.c
$ ex5

```

.....



```

kyumin@DESKTOP-NUDFAPK ~
$ ex5
hello: 1900 0
hello: 1900 1901
hello: 0 1902
hello: 0 0

```

총 4줄이 출력되는데 그 이유는

7) Try below (ex6.c). How many hellos do you see? Explain why you have that many hellos.

```
void main(){
    int x,y;
    x=fork();
    printf("hello: %d\n", x);
    y=fork();
    printf("hello: %d\n", y);
}
```

```
$ gcc -o ex6 ex6.c
```

```
$ ex6
```

```
.....
```

8) Try below (ex7.c). When you run ex7, how many processes run at the same time? Which process finishes first and which process finishes last? Show the finishing order of the processes. Run ex7 again and compare the finishing order with that of the first run. If different, explain why.

```
#include <stdio.h>
#include <stdlib.h>
#include <sys/types.h>
#include <unistd.h>
void main(){
    int x, i;
    for(i=0;i<5;i++){
        x=fork();
        if (x==0){ // child
            int k;
            for(k=0;k<1000;k++){
                printf("%d-th child running %d-th iteration\n", i, k);
                fflush(stdout);          // to make printf work immediately
                usleep(10000);           // sleep 10000 microseconds
            }
            exit(0);  // child exits after 1000 iterations
        }
    }
    // now parent
```

```
    printf("parent exits\n");  
}
```

9) If you delete "exit(0)" in ex7.c, how many processes will be created? Confirm your answer by modifying the code such that each process displays its own pid. You may want to decrease the size of inner loop (k loop) to "k<10" to shorten the run time.

10) Write a program that creates n child processes where n is specified by the user. Let each process prints some message when it has finished.

```
$ ex1  
how many child processes?  
5  
child 1 finished  
child 2 finished  
child 4 finished  
child 3 finished  
child 5 finished  
parent finished
```

11) Repeat Problem 10, but let the processes end in the reverse order in their creation time as shown below. Use sleep() function to force the order.

```
$ ex1  
how many child processes?  
5  
parent finished  
child 5 finished  
child 4 finished  
child 3 finished  
child 2 finished  
child 1 finished
```

12) Write a program that the parent writes some 1-digit number x into a file and the child reads it and prints x + 1. Assume x is less than 10. Check this file with xxd if it has the number written by the parent.

```
$ ex1
```


parent writes 7

child prints 8