signal

lec

(Do not use "^z" when experimenting with signals. ^z will stop the process, and the

process cannot receive further signals.)

3번 터미널을 죽이라는 것은 putty 창을 닫으라는 겁니다(x 표 클릭해서)

putty 창을 닫으면 putty 창을 제어하는 shell을 죽이게 되고 shell이 죽으면 이 shell 밑에서 생성된 모든 프로세스에 signal 1번이 전달됩니다. 따라서 시그널 1번에 대비하지 않은 프로세스들은 모두 죽게 됩니다.

ps 에서 defunct 로 나오는 프로세스는 좀비프로세스입니다. 프로세스가 exit하면 일단 좀비가 되어 바디는 없어지지만 프로세스 디스크립터는 시스템에 남아있습니다. 이 상태를 좀비라고 부릅니다.

이 좀비는 부모가 wait을 호출할 때까지 시스템에 남아있습니다. 8번 숙제는 이 좀비를 어떻게 없앨 수 있는가에 대한 것입니다.

좀비는 이미 죽었으므로 kill -9 로도 죽일 수 없습니다 (죽었는데 또 죽일 수 없습니다). 부모가 wait 을 호출해야 사라집니다. 8번은 이 좀비의 부모가 어떻게 wait을 호출할 수 있도록 할 것인가 하는 문제입니다. 9번은 아예 좀비가 안 생기게 하려면 어떻게 하는가 하는 문제입니다. 좀비는 8192 바이트(프로세스 디스크립터 크기) 메모리를 차지하므로 좀비가 많아지면 시스템 성능이 다운됩니다. 따라서 프로그래머는 자신의 프로그램이 좀비를 생성하지 않도록 주의해야 합니다.

8번 해답은 부모를 죽이는 겁니다.

부모를 죽이면 자식이 1번에게 입양이 되는데 1번은 강의노트/ppt 에 나와있듯이 무한 루프를 돌면서 wait을 계속 호출하고 있는 프로세스입니다.

내일 시험과 관련해서 채팅서버 문제가 하나 반드시 나오니 준비하기 바랍니다. 샘플로 올린 김하림양의 코드를 이해하기 바랍니다. 채팅서버의 프로토콜을 변경하는 문제가 나올 예정입니다. 다른 문제는 아직 생각 중입니다.

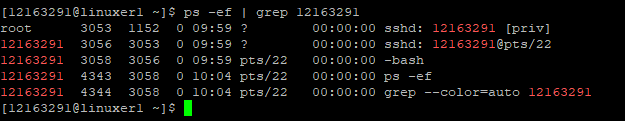
시험은 9-11 2시간입니다

hw

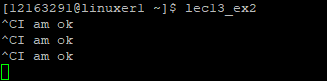
1) Try to kill lec13\_ex1 and lec13\_ex2 in Section 1 with ^c. What happens? You cannot kill lec13\_ex2 with ^c, just leave it and go to homework 1-1).



lec13\_ex1 + Ctrl C



종료된게 맞습니다.



lec13\_ex2 + Ctrl C

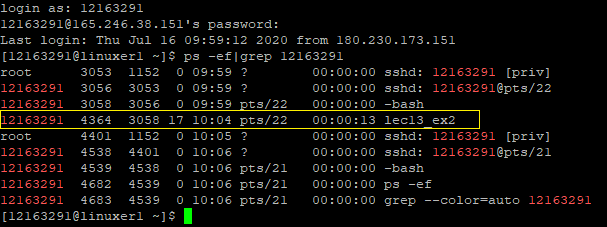
1-1) Open another putty window and find the pid of lec13\_ex2 by

$ ps -ef | grep 12345

where 12345 is your student ID. Suppose lec13\_ex2's pid is 334455. Send SIGINT (signal number 2) to ex2 with

$ kill -2 334455

What happens? Explain the result.



다른 putty 창에서 PID를 출력했습니다.

: 4364

|  |  |
| --- | --- |
|  |  |
| ex2 | 다른 window |

프로세스가 kill되지 않고, foo() 내용이 출력되는 것을 확인할 수 있습니다.

2) You can kill lec13\_ex2 as follows:

open another terminal

find the pid of ex2, e.g. 1234

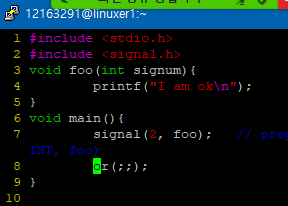
kill 1234 (or “kill -15 1234”)

Why SIGINT (signal 2) cannot kill lec13\_ex2 while SIGTERM (signal 15) can?

|  |  |
| --- | --- |
|  |  |
| ex2 | 다른 window |

kill -2 가 아닌, kill -15를 통해

lec13\_ ex2를 terminate할 수 있었습니다.



signal option 15에 대한 코드가 없기 때문에 위의 과정이 가능합니다.

2-1) Change lec13\_ex2.c as follows.

#include <stdio.h>

#include <signal.h>

void foo(int signum){

printf("I am ok\n");

}

void main(){

signal(2, foo); // prepare for signal number 2. same as signal(SIGINT, foo)

signal(15, foo);

for(;;);

}

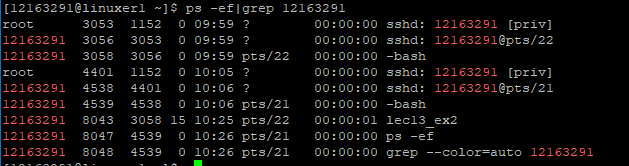
//kill 되지 않고 foo()가 실행된다

Now compile and run lec13\_ex2 and try to kill it with SIGINT and SIGTERM in another window. Assume lec13\_ex2 has pid 334455.

$ kill -2 334455

$ kill -15 334455

Explain the result.



PID ; 8043

|  |  |
| --- | --- |
|  |  |
| ex2 | 다른 window |

signal로 -2, -15를 만날경우

ok를 출력하도록 해놓았기 때문에,

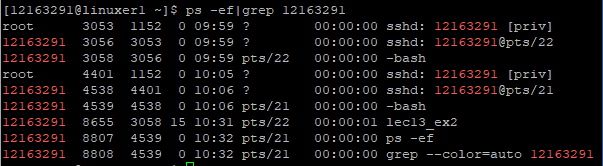
iam ok가 두번 출력됩니다.

|  |  |
| --- | --- |
|  |  |
| ex2 | 다른 window |

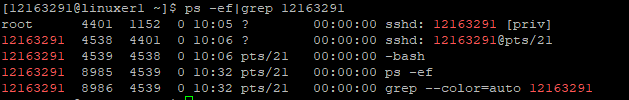
-2, -15가 아닌 다른옵션값 -1 로 kill해주었습니다.

그러자 프로세스가 끝이 났습니다.

3) You can also kill lec13\_ex2 by closing the terminal where lec13\_ex2 is running. (창 닫기) Confirm this by first finding out the pid of lec13\_ex2 with "ps –ef | grep 1234" (assuming your student id is 1234) in another terminal, close the terminal where lec13\_ex2 is currently running, and find this pid again with "ps –ef | grep 1234" in the second window. Explain why lec13\_ex2 dies in this case.

PID : 8655

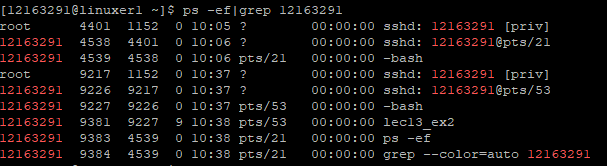
다른창을 닫은 후



lec13 \_lec2 process가 종료된 것을 확인할 수 있습니다

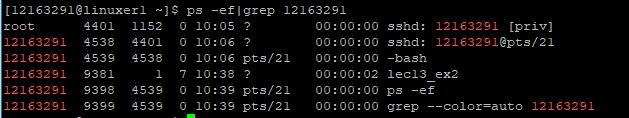
4) How can you prevent lec13\_ex2 from being killed even when you close the terminal?

signal ( window종료로 인한 signal#, foo); 를 통해 막을 수 있습니다



PID : 9381

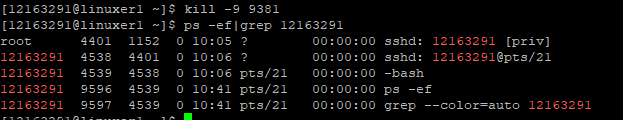
다른 window창을 닫은후,



root 가 하나인 것으로 window는 하나만 열려있는 것을 알 수 있습니다.

그러나 lec13\_ex2는 종료되지 않은 것을 확인할 수 있었습니다.

5) If you were successful in problem 4 in preventing lec13\_ex2 from being killed, how can you still kill lec13\_ex2?



아직 signal()로 대비되어있지 않던, -9 옵션을 통해 프로세스를 kill해주었습니다.

6) Write a program (lec13\_ex3.c) that forks two children as below. All three processes (the parent and the two children) run infinite loops. Run this program and find t15he pids of the three processes in another window. Kill the parent ("kill -15 parent-pid") and observe what happens to the children.

..................

x = fork();

if (x==0) for(;;); // first child

else{

y=fork();

if (y==0) for(;;); // second child

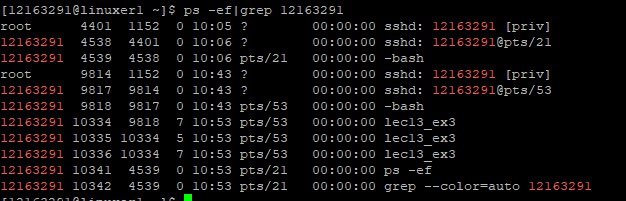
}

for(;;); // parent

...........

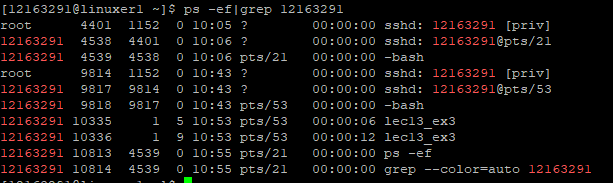
다른 창에서 ps-ef로 PID찾아서

최상위부모 -15로 KILL하고 그 결과 설명



parent PID : 10334

|  |  |
| --- | --- |
|  |  |
| lec13\_ex3 | 다른 window |



parent만 kill되었습니다.

children은 kill되지 않았습니다.



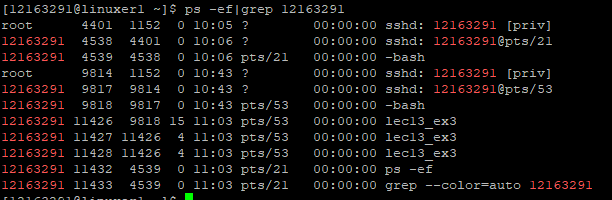
모두 kill해주었습니다.

7) Same as 6), but this time kill the shell of the terminal where lec13\_ex3 is running with

"kill -9 114455", where 114455 is the pid of the shell.

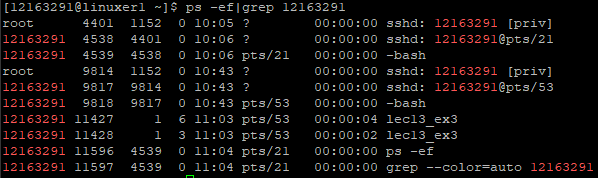
What happens to the three processes (the parent and the two children)?

Why the three children die and how can you prevent the three processes from being killed?



parent PID : 11426

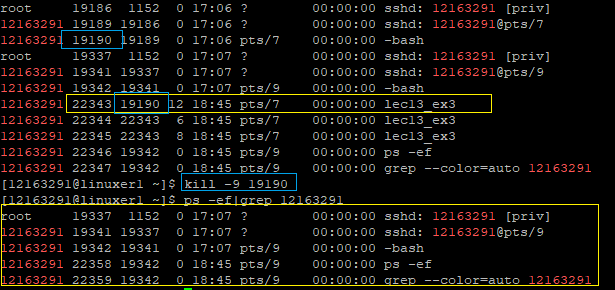
|  |  |
| --- | --- |
|  |  |
| lec13\_ex3 | 다른 window |



자식들은 남아있었습니다.

parent pid는 1이 되었습니다.

+



parent의 ppid를 -9로 kill하게되면

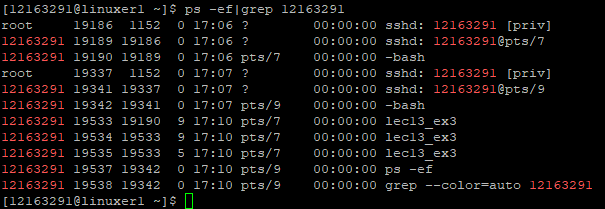
다른 window창이 꺼지며, 한번에 모든 3개의 process가 kill됩니다.

8) Same as 6), but this time kill one of the children instead of the parent. What happens to the dead child?

Can you remove this dead child (zombie) with signal 9? NO

How can you remove this dead child? 부모가 wait()

8번 해답은 부모를 죽이는 겁니다

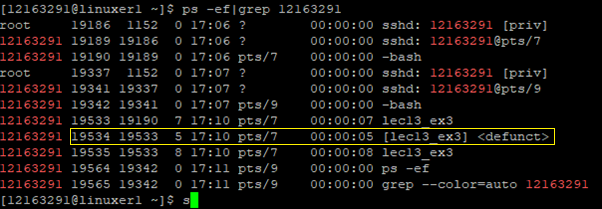


parent PID: 19533

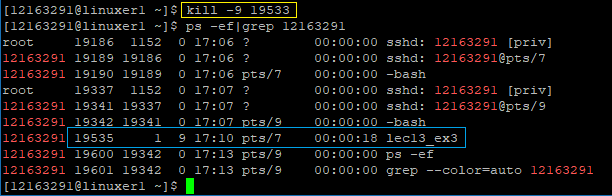
children PID : 19534, 19535



signal 9로kill할수 없었습니다. (아래의 결과를 보면 알 수 있습니다.)



<defunct> = zombie가 되었습니다.



parent PID를 -9로 kill해주니 <defunct> 좀비도 kill된 것을 확인할 수 있었습니다.

9) In problem 8), how can you prevent the generation of a zombie child?

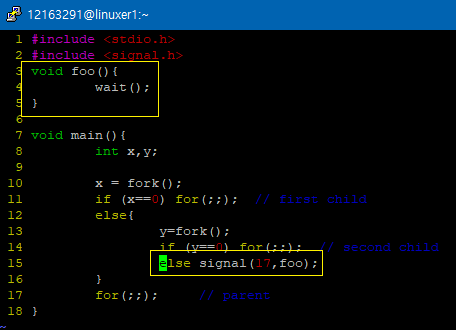
Modify your code in 6) to prevent the generation of zombie child.

Remember the parent cannot wait in "wait()" since the parent should run its infinite loop too.

The parent should prepare a handler for "**SIGCHLD**" signal and call "wait()" inside this handler.



를 사용해줍니다.



the parent cannot wait in "wait()" since the parent should run its infinite loop too

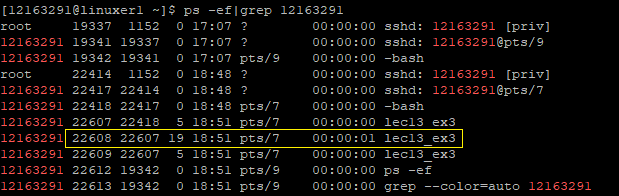
이기 때문에, 17이라는 – child가 die하면 반응하는 signal( )함수를 생성후,

그 결과로 wait()을 하도록 해줍니다.

이는 가장 안쪽의 if문의 else문에 넣어줍니다.

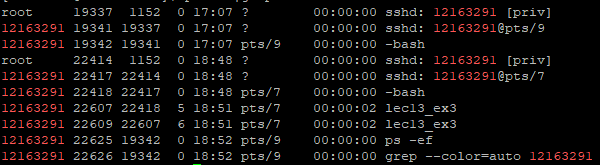
child가 kill되면서 signal을 만나 wait하게 될 것입니다.

그렇게 되면 자식이 좀비가 되는 것을 막을 수 있습니다.



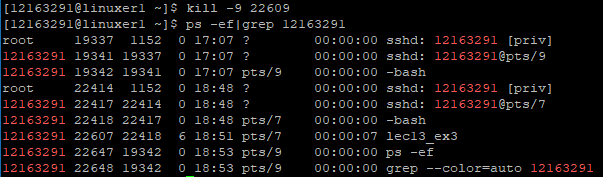


8)번에서와 같이, 자식중 하나를 -9로 kill해주었습니다



zombie가 되지않고

잘 kill된 것을 확인할 수 있습니다.



남은 자식도 kill해줍니다.