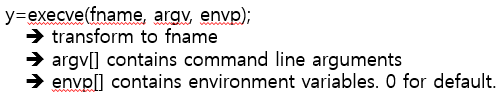
lec

exec = transform

process는 그대로인데 실행되는 것이 달라진 것 – 가리키는 mm만 달라짐

exec를 사용한 코드는 사라지고, 해당 프로세스는 exec로 호출한 새 코드의 메모리를 가리키게 된다.



두번째인자가 중요



절대 full경로입력 해야한다

4번은 exec 다음에 printf("Job done\n") 하는 것은 의미가 없습니다. exec 에 성공하면 현재 프로그램은 지워지고 exec을 요청한 프로그램이 덮어쓰기 때문입니다. 즉 exec 다음에 나오는 문장들은 모두 사라집니다. 결국 exec을 해서 프로그램이 사라져도 여전히 printf("Job done\n")을 할 수 있는 프로세스를 사전에 fork를 통해 만들어 두어야 합니다. 그리고 이 프로세스는 exec을 하는 프로세스가 종료된 이후에 실행되어야 하므로 sleep(1)으로 1초간 기다리게 하든지 아니면 wait()으로 exec을 하는 자식 프로세스가 종료되기를 기다리게 해야 합니다.

5번은 cat 명령어만 실행하는 것이 아니고 어떤 명령어든지 실행할 수 있어야 합니다. 즉 command를 받아서 strtok로 각 인자를 잘라낸 후 argv 배열에 담아서 exec을 해야 합니다.

hw

1-1) Try below and explain why the output is "I am ex1" when ex2 runs.

ex1.c:

#include <stdio.h>

void main(){

printf("I am ex1\n");

}

ex2.c:

#include <stdio.h>

void main(){

execve(“./ex1”,0 , 0); // change to ./ex1 with no additional argument

printf("I am ex2\n");

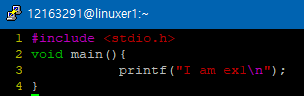
}

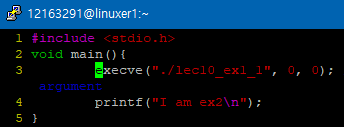
$ gcc -o ex1 ex1.c

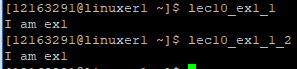
$ gcc -o ex2 ex2.c

$ ex2

I am ex1







ex2를 실행하였지만, 그 안의 execve 함수로 인해 ex2가 ex1으로 transform되어

“I am ex1”이 출력됩니다.

execve 함수가 성공하면 그 아래의 코드는 사라집니다. 때문에 “I am ex2”는 출력되지 않습니다. execve의 실행이 실패하였을 경우에는 “I am ex2”만 출력됩니다.

\* 이름을 똑같이 입력해야함에 유의해야 합니다.

1-2) Run myexec below. Explain the result.

myexec.c:

#include <stdio.h>

void main(){

char \*k[10];

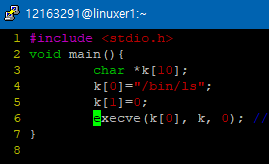
k[0]=”/bin/ls”;

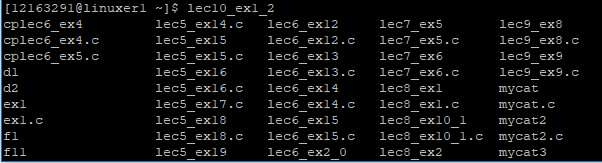
k[1]=0;

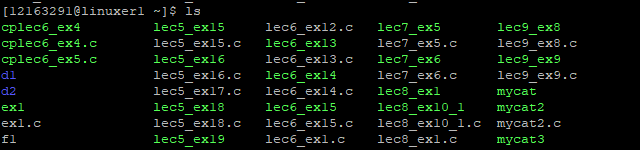
execve(k[0], k, 0); // change to /bin/ls with no additional argument

}

The above program will exec to /bin/ls and print the listing of files in the current directoy.







ls 커맨드의 결과와 같은 것을 확인할 수 있었습니다.

lec10\_ex1\_2.c는 execve 함수를 사용하였는데, target program으로 “/bin/ls”를 입력하였기 때문에 lec10\_ex1\_2을 실행시키면 ls로 바뀌어 실행됩니다.

이 과정에서 추가적인 argument는 없었습니다.

1-3) Run myexec below and explain the result.

myexec.c:

void main(){

char \*x[10];

x[0]="/bin/cat";

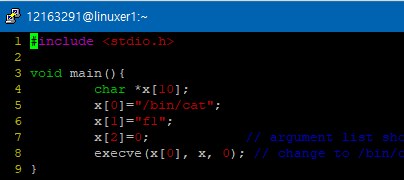
x[1]="f1";

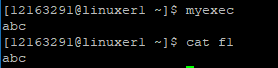
x[2]=0; // argument list should end with 0

execve(x[0], x, 0); // change to /bin/cat with one argument f1

}

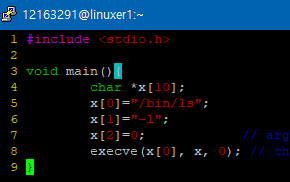
The above program will exec to "/bin/cat f1" which will print the contents of f1.

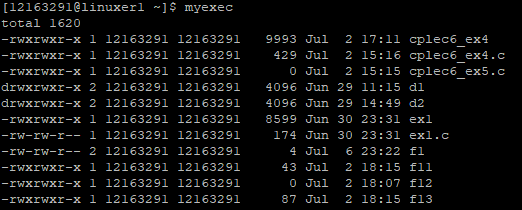


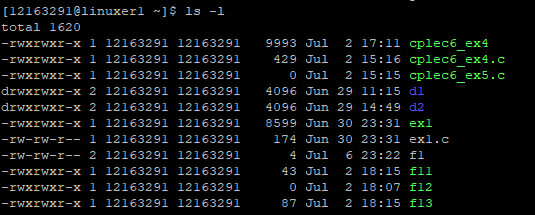


1-2) 번과 동일한 mechanism 입니다.

2) Change myexec such that it execs to “/bin/ls -l”.







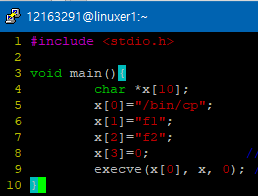
결과는 같습니다.

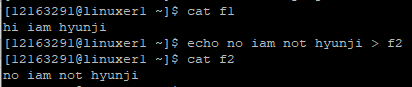
execve 함수의 target program은 “/bin/ls”이고,

이번에는 argument로 옵션 ”-l ”도 입력하였습니다.

그 결과 myexec를 실행하면 ‘ls –l’과 같은 결과가 나오는 것을 확인할 수 있었습니다.

3) Change myexec such that it execs to “/bin/cp f1 f2”.





f1 내용 = hi iam hyunji

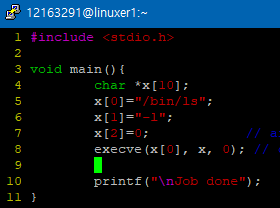
f2 내용 = no iam not hyunji

myexec를 실행시켜 보겠습니다.



f2의 내용이 f1으로 바뀜 (cp가 성공적으로 됐음) 을 확인할 수 있었습니다.

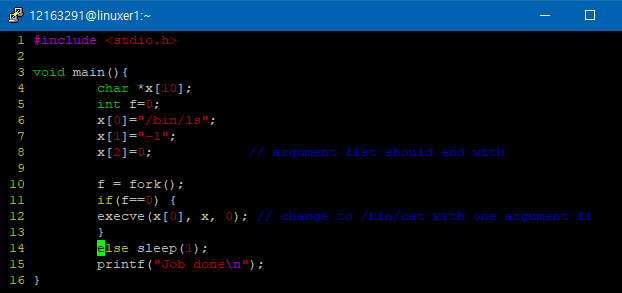
4) Change myexec such that it runs “/bin/ls -l” and prints “job done” after “/bin/ls -l” is finished.



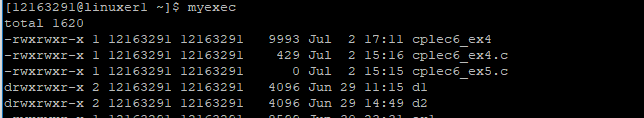
이렇게 하면 안되는 이유는, exec 함수가 실행되면 해당 프로그램은 삭제되기 때문입니다.

fork사용하는게 낫나? 맞음!

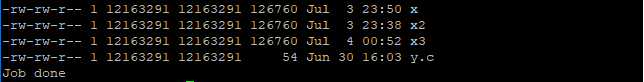
대신 sleep(초), wait()함수등을 사용하여 ls -l프로그램이 끝날때까지 자식 프로세스가 기다릴수 있도록 해야합니다.



수정된 코드의 내용은 위와 같습니다.



…



ls -l의 결과가 모두 출력된 후,

1초뒤에 ( sleep(1)에 의해 )

Job done이 출력됩니다.

> f(fork의 rtn값) 가 0이면 chile process입니다. 이때에 exec함수가 실행되도록 하였습니다.

execve 함수는 실행되면, 이후의 코드가 실행되지 않습니다.

> f가 0이아니면 parent process 이고, 이때 sleep을 통해 child 가 먼저 끝나기까지 1초 기다리도록 하였습니다.

그후, 코드를 넣어주면 그 코드가 실행되게 됩니다.

5) Change myexec such that it reads a command and execs to the given command.

$ myexec

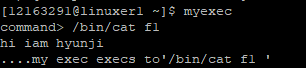
command> /bin/cat f1

….myexec execs to "/bin/cat f1"

$ myexec

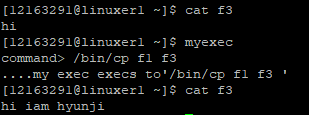
command> /bin/cp f1 f2

….myexec execs to "/bin/cp f1 f2"





잘 실행되는 것을확인할 수 있었습니다.



cp가 잘 되었나 확인하여 보니 잘 된 것을 확인할 수 있었습니다.

사용된 코드는 다음과 같습니다.



우선, string을 사용해야하므로 헤더를 추가해줍니다.

gets로 char값을 입력 받습니다.

그 char값 ( 순서대로 들어온 알파벳들)에 “ ”기준으로 tokenize를 실행하였습니다.

그 후 그 결과를 x[]에 입력하여 x[]에는 program 이름과 그 외 argument값들이 “ ”으로 나눠진 결과로써 각각 저장됩니다

6) Same as 5), but myexec will repeat the process 5 times.

(\* to avoid system-down, use a fixed-iteration loop:

for(;;){…} 🡺 for(i=0;i<5;i++){…}

\*)

$ myexec

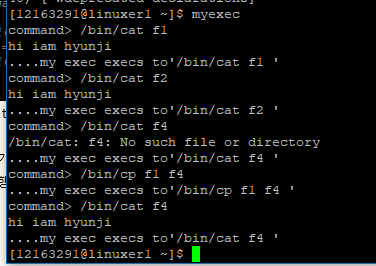
command> /bin/cat f1

….display the contents of f1

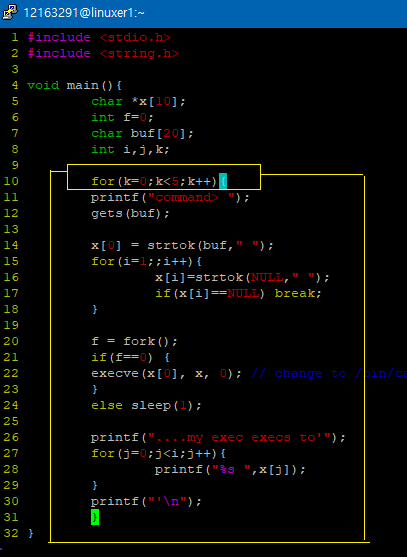
command> /bin/ls

….display file names in the current directory

………….



총 다섯번의 실행을 한 후, 종료되며 커맨드라인이 뜨는 것을 확인할 수 있습니다.



5)의 코드와 거의 동을하지만, for문을 통해 5번 반복을 해주었습니다.

7) Same as 6), but change the prompt to the current location and ‘$’ as follows. You may need “getcwd”.

$ myexec

[/home/sp1/12345]$ /bin/cat f1

….display the contents of f1

[/home/sp1/12345]$ /bin/ls

….display file names in the current directory

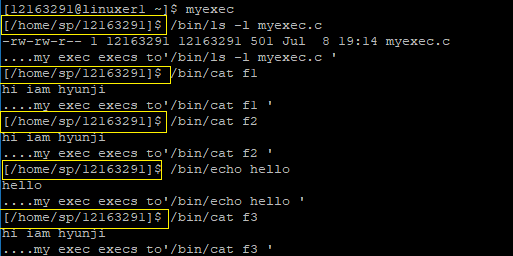
………….

getcwd 함수를 사용하면, 현 위치를 정보를 불러와 사용할 수 있습니다.

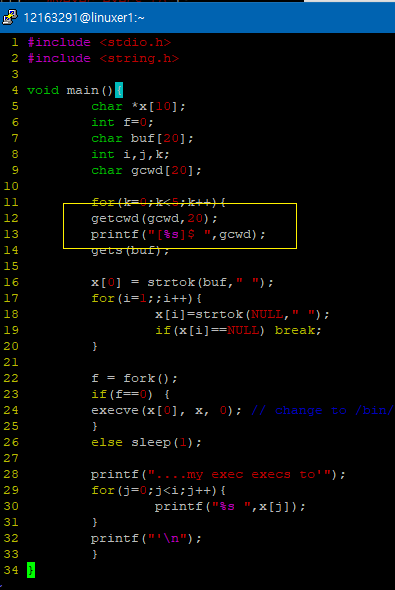
(pwd 같은 현 정보)



처음에는 cwd정보를 저장할 변수를, 두번쨰 인자에는 그 변수의 크기를 입력 합니다



6)번과 동일하지만, command 와 달리, 현재 제 pwd 값이 출력되고 있는 것을 확인할 수 있었습니다.



getcwd 함수를 사용하여 현 위치 정보를 얻습니다.

20개의 string을 gcwd라는 character형 array에 저장하고,

그것들을 printf 함수를 사용하여 출력하였습니다. - pwd정보 출력