

## Lecture 10: Exec, Exit, Wait

1. exec : execve, execl, execlp, ....

A program calls exec to transform itself to another.

1) algorithm of exec

- remove old body
- load new body
- adjust process descriptor

2) function prototype of execve

```
y=execve(fname, argv, envp); // change to fname with additional arguments specified in
                             // argv[] and additional environment variables specified in
                             // envp[]. returns -1 if error.
```

```
y=execve("/aa/bb", k, 0); // change to /aa/bb with additional arguments in k
```

## 2. Homework

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>
#include <unistd.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
```

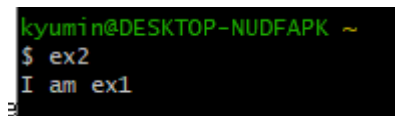
```
$ ex2
```

```
I am ex1
```

For Cygwin, the above code will not work. Change as below:

ex2.c:

```
#include <stdio.h>
#include <unistd.h>
void main(){
    char *argv[5];
    argv[0]="./ex1";
    argv[1]=0;
    execve(argv[0], argv, 0); // change to ./ex1 with no additional argument
    printf("I am ex2\n");
}
```

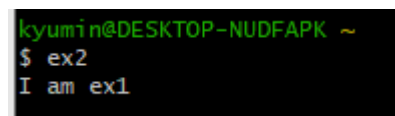


```
kyumin@DESKTOP-NUDFAPK ~
$ ex2
I am ex1
```

Ex2 프로그램을 실행 중 execve가 실행된다면 ex2의 바디가 지워지고, ex1이 메모리에 로드되므로 ex1 프로그램이 실행된다.

To check whether execve succeeds or not:

```
#include <stdio.h>
#include <unistd.h>
void main(){
    char *argv[5];
    argv[0]="./ex1";
    argv[1]=0;
    int y=execve(argv[0], argv, 0); // change to ./ex1 with no additional argument
    if (y < 0){                // we have an error in execve
        perror("err:");        // show the reason of error
        exit(0);
    }
    printf("I am ex2\n");
}
```



```
kyumin@DESKTOP-NUDFAPK ~
$ ex2
I am ex1
```

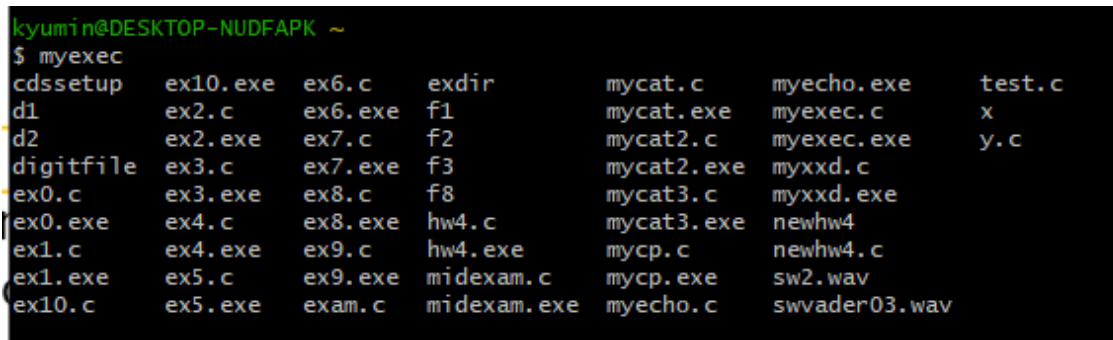
Execve가 실행이 되면서 ex2의 바디는 제거가 되버리고, ex1이 실행이 되면서 “I am ex1”이 출력이 됨.

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

myexec.c:

```
#include <stdio.h>
#include <unistd.h>
void main(){
    char *k[10];
    k[0]="/bin/ls";
    k[1]=0;        // make sure you put 0 at the end
    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.



```
kyumin@DESKTOP-NUDFAPK ~
$ myexec
cdssetup  ex10.exe  ex6.c  exdir  mycat.c  myecho.exe  test.c
d1        ex2.c   ex6.exe f1      mycat.exe myexec.c    x
d2        ex2.exe  ex7.c  f2      mycat2.c  myexec.exe  y.c
digitfile ex3.c   ex7.exe f3      mycat2.exe myxxd.c
ex0.c     ex3.exe  ex8.c  f8      mycat3.c  myxxd.exe
ex0.exe   ex4.c   ex8.exe hw4.c   mycat3.exe newhw4
ex1.c     ex4.exe  ex9.c  hw4.exe mycp.c   newhw4.c
ex1.exe   ex5.c   ex9.exe midexam.c mycp.exe  sw2.wav
ex10.c    ex5.exe  exam.c  midexam.exe myecho.c  swvader03.wav
```

프로그램 실행 시 ls가 실행된다. 현재 위치의 파일과 디렉토리가 출력되는 것을 볼 수 있다.

1-3) Run myexec below. Explain the result.

myexec.c:

```
#include <stdio.h>
#include <unistd.h>
void main(){
    char *k[10];
    k[0]="/bin/ls";
    k[1]="-l";      // l is small L not number 1
    k[2]=0;         // make sure you put 0 at the end
    execve(k[0], k, 0); // change to "/bin/ls -l"
}
```

The above program will exec to "/bin/ls -l" and print a long listing of files in the current directoy.

```

kyumin@DESKTOP-NUDFAPK ~
$ myexec
total 1594
drwx-----+ 1 kyumin 없 음      0 Mar 15  2021 cdssetup
drwxr-xr-x+ 1 kyumin 없 음      0 Apr 14 15:45 d1
drwxr-xr-x+ 1 kyumin 없 음      0 Mar  6 12:47 d2
-rwxr-xr-x+ 1 kyumin 없 음      1 Apr 18 23:21 digitfile
-rw-r--r--+ 1 kyumin 없 음    249 Apr 15 09:10 ex0.c
-rwxr-xr-x+ 1 kyumin 없 음   66203 Apr 15 09:11 ex0.exe
-rw-r--r--+ 1 kyumin 없 음     60 Apr 29 09:04 ex1.c
-rwxr-xr-x+ 1 kyumin 없 음   66542 Apr 29 09:02 ex1.exe
-rw-r--r--+ 1 kyumin 없 음     566 Apr 18 23:21 ex10.c
-rwxr-xr-x+ 1 kyumin 없 음   68716 Apr 18 23:21 ex10.exe
-rw-r--r--+ 1 kyumin 없 음     361 Apr 29 09:23 ex2.c
-rwxr-xr-x+ 1 kyumin 없 음   67119 Apr 29 09:26 ex2.exe
-rw-r--r--+ 1 kyumin 없 음     177 Apr 15 09:54 ex3.c
-rwxr-xr-x+ 1 kyumin 없 음   66007 Apr 15 09:54 ex3.exe
-rw-r--r--+ 1 kyumin 없 음     193 Apr 15 10:03 ex4.c
-rwxr-xr-x+ 1 kyumin 없 음   66007 Apr 15 10:03 ex4.exe
-rw-r--r--+ 1 kyumin 없 음     150 Apr 15 10:07 ex5.c
-rwxr-xr-x+ 1 kyumin 없 음   66007 Apr 15 10:07 ex5.exe

```

파일 이름뿐만 아니라 추가 argument로 “-l”이 들어왔다. 그래서 ls 실행 시 더 자세한 정보가 출력되는 것을 볼 수 있다.

1-4) 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.

```

kyumin@DESKTOP-NUDFAPK ~
$ myexec
I have a dream
that one day this nation
will rise up and
live out the true
meaning of its creed
that all men are created equal.

kyumin@DESKTOP-NUDFAPK ~
$ cat f1
I have a dream
that one day this nation
will rise up and
live out the true
meaning of its creed
that all men are created equal.

```

추가 argument로 “f1”이 들어왔다. 그래서 f1의 파일 내용이 출력되는 것을 볼 수 있다. Cat 명령어로 출력해본 결과와 비교해보면 동일하게 출력된 것을 볼 수 있다.

2) Change myexec such that it execs to “/bin/ls -l -a”. Note “l” is small L, not number 1.

```

#include <stdio.h>
#include <unistd.h>
void main(){
    char *x[10];
    x[0]="/bin/ls";
    x[1]="-l";
    x[2]="-a"; // argument
    x[3]=0;;
    execve(x[0], x, 0); // c
}

```

```

kyumin@DESKTOP-NUDFAPK ~
$ myexec
total 1658
drwxrwx---+ 1 Administrators  0 Apr 29 09:39 .
drwxrwx---+ 1 kyumin         0 Apr 27 17:08 ..
-rw-----+ 1 kyumin         6001 Apr 26 20:49 .bash_history
-rw-r--r--+ 1 kyumin         49 Apr 7 18:03 .bash_profile
-rw-r--r--+ 1 kyumin         20 Apr 7 18:01 .bashrc
-rw-r--r--+ 1 kyumin         0 Apr 23 14:59 .chapttest
-rwx-----+ 1 kyumin         51 Nov 16 2021 .gitconfig
drwx-----+ 1 kyumin         0 Jul 15 2021 .idlerc
-rw-----+ 1 kyumin         20 Apr 26 20:49 .lessht
-rw-----+ 1 kyumin        12288 Mar 1 17:49 .test.c.swp
-rw-----+ 1 kyumin        16667 Apr 29 09:39 .viminfo
drwx-----+ 1 kyumin         0 Mar 15 2021 cdssetup
drwxr-xr-x+ 1 kyumin         0 Apr 14 15:45 dl
drwxr-xr-x+ 1 kyumin         0 Mar 6 12:47 d2
-rwxr-xr-x+ 1 kyumin         1 Apr 18 23:21 digitfile
-rw-r--r--+ 1 kyumin         249 Apr 15 09:10 ex0.c
-rwxr-xr-x+ 1 kyumin        66203 Apr 15 09:11 ex0.exe
-rw-r--r--+ 1 kyumin         60 Apr 29 09:04 ex1.c
-rwxr-xr-x+ 1 kyumin        66542 Apr 29 09:02 ex1.exe

```

추가 argument로 “- l”과 “-a”를 추가해줬다. 프로그램을 실행해보면 “ls -l -a”명령어를 실행한 것과 같은 결과가 나온다.

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

```

kyumin@DESKTOP-NUDFAPK ~
$ myexec

kyumin@DESKTOP-NUDFAPK ~
$ cat f1
I have a dream
that one day this nation
will rise up and
live out the true
meaning of its creed
that all men are created equal.

kyumin@DESKTOP-NUDFAPK ~
$ cat f2
I have a dream
that one day this nation
will rise up and
live out the true
meaning of its creed
that all men are created equal.

```

Cp f1 f2와 동일한 결과가 나온다.

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

5) Change myexec such that it reads a command and execs to the given command. Your code should be able to exec any command. Read the command line with gets() or fgets() and use strtok() to extract all arguments and perform exec to run the command. Remember you have to remove the last character('\n') if you use fgets.

```

$ myexec
command> /bin/cat f1
....myexec execs to "/bin/cat f1"
$ myexec
command> /bin/cp f1 f2
....myexec execs to "/bin/cp f1 f2"

```

```

#include <stdio.h>
#include <unistd.h>
#include <string.h>

void main(){
    char *x[10];

    char pattern[100];
    fgets(pattern, 100, stdin);
    pattern[strlen(pattern) - 1] = 0;
    x[0] = strtok(pattern, " ");
    for(int i = 1;; i++){
        x[i] = strtok(NULL, " ");
        if(x[i] == NULL) break;
    }

    execve(x[0], x, 0); // change to /bin/
}
~

```

```

kyumin@DESKTOP-NUDFAPK ~
$ myexec
/bin/cat f1
I have a dream
that one day this nation
will rise up and
live out the true
meaning of its creed
that all men are created equal.

kyumin@DESKTOP-NUDFAPK ~
$ myexec
/bin/cp f1 f2

```

Fget로 입력을 받는 경우 마지막 문자에 개행 문자가 들어가므로, 0으로 초기화해줬다. 그리고 strtok 함수를 사용해 x에 단어별로 저장해줬다. Execve에서 입력된 x의 명령어를 실행해준다.

6) Same as 5), but myexec will repeat the process forever until you stop the program with “ctrl-c”. Also display the result of strtok as shown below.

```

$ myexec
command> /bin/cat f1
argc: 2 argv[0]: /bin/cat argv[1]: f1 argv[2]: 0
....display the contents of f1
command> /bin/cp f1 f2
argc: 3 argv[0]: /bin/cp argv[1]: f1 argv[2]: f2 argv[3]: 0
....copy f1 to f2

```

```
command> /bin/cat f2
```

```
.....
```

7) Same as 6), but change the prompt to the current location and '\$' as follows. You may need "getcwd". Also add code to detect "exit" command and stop the program when the user types "exit".

```
$ myexec
[/home/sp1/12345]$ /bin/cat f1
argc: 2 argv[0]: /bin/cat argv[1]: f1 argv[2]: 0
....display the contents of f1
[/home/sp1/12345]$ /bin/cp f1 f2
argc: 3 argv[0]: /bin/cp argv[1]: f1 argv[2]: f2 argv[3]: 0
....copy f1 to f2
[/home/sp1/12345]$ /bin/cat f2
.....
[/home/sp1/12345]$ exit
bye
$
```

#### 4. exit, wait

exit: A program calls exit() to exit.

- remove the body
- becomes a zombie until the parent calls wait()

wait: A program calls wait() and waits for the child to exit.

- if the child already called exit()
  - remove its process descriptor
- else
  - wait until the child exits

#### 5. shell

algorithm:

```
for(;;){
    printf("$");
    scanf("%s", buf);
    x=fork();
    if (x==0) execve(buf,0,0);
```



```
        else wait();
    }
}
```

Actual code: mysh.c

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

void main(){
    int x,y,status, i;
    char buf[50];
    char * argv[10];

    for(i=0;i<10;i++){ // use a finite loop instead of an infinite one
        printf("$");
        scanf("%s", buf); // get command.
        argv[0]=buf;
        argv[1]=0;

        x=fork();
        if (x==0){ // child
            printf("I am child to execute %s\n", buf);
            y=execve(buf, argv, 0);
            if (y<0){
                perror("exec failed");
                exit(1);
            }
        } else wait(&status);
    }
}
```

## 6. process tree

(For Cygwin, use “ps -W” to see all processes including window processes. To see ppid in windows, download “process explorer” and run it.)

```
kernel (pid=0)
fork: init (pid=1)
exec: /sbin/init (pid=1)
fork & exec: iscsid
      rsyslogd
      .....
      /usr/sbin/sshd (pid=1262)
fork : sshd : linuxer (pid=5198)
fork : sshd: linuxer@pts/0 (pid=5201)
fork & exec: -bash (pid=5202)
fork & exec: vi cli.c (pid=6420)
```

## 7. debugging a program with fork

In gdb, use "set follow-fork-mode child" or "set follow-fork-mode parent" to debug child or parent process. (Cygwin does not support "set follow-fork-mode". Just debug up to "fork" in cygwin.)

## 8. Homework

1) Compile and run mysh.c in section 5. What is the difference between mysh and the system shell(the login shell that runs when you log in)? Show at least 5 differences.

2) What is the process name of your login shell? What is the executable file name of your login shell and how can you find it? Who is the parent of your login shell? Explain how the parent of your login shell can create your login shell by showing its C code(roughly). Find all ancestor processes of your login shell.

3) (Builtin Command) Improve mysh such that it exits when the user types "exit". You have to handle "exit" before "fork". Explain why. This kind of commands that the shell has to handle before fork are called built-in commands.

4) Improve mysh further such that it can handle a command with arguments, such as "/bin/ls -l". Use gets() or fgets() to read the command.

4-1) Improve it further so that it can handle "cd" comand. Also improve it so that it can handle "pwd" command. Note "cd" and "pwd" are other examples of built-in command.

5) (Handling &) Change the shell such that it can handle '&' at the end of the command.

```
$ ex1
```

In above, the shell waits until ex1 (the child) is finished. You should make ex1 to have an infinite loop to see the effect.

```
$ ex1 &
```

In above, the shell does not wait and immediately prints the next prompt and waits for the next user command. Make sure you delete "&" at the end of the command once you detect it.

6) (Handling relative path) Make your shell handle relative paths assuming the executable file always exists in /bin directory. When the user enters only the command name (e.g. "ls -l", "cp f1 f2", etc), build a full path such as "/bin/ls", "/bin/cp", etc. and perform exec. Use sprintf() to build the full path.

6-1) Use getenv("PATH") to retrieve PATH environment variable and use strtok() to extract each system path. Display each system path line by line.

```
/usr/lib64/ccache
/usr/local/bin
/usr/bin
.....
```

7) (Handling relative path) Change the shell such that it can handle relative path for the command in general. The shell will search the PATH environment variable to compute the full path of the command when it is given as a relative path name. Use getenv("PATH") to obtain the pointer to the value of the PATH environment variable. Note you need to copy the string of the PATH variable into another char array before you start extracting each path component with strtok() since strtok() destroys the original string.

8) dup(x) duplicates fd[x] in the first empty entry in the fd table. Run following program and explain the output. Assume f1 has

```
hello my boy
```

```
x=open("f1", O_RDONLY, 00777);
int y;
y=dup(x);
printf("x:%d y:%d\n", x, y);
char buf[50];
int k=read(x, buf, 5);
buf[k]=0;
printf("buf:%s\n", buf);
k=read(y, buf, 5);
buf[k]=0;
printf("buf:%s\n", buf);
```

9) (Standard output redirection) Explain the output of the following code.

```
x=open("f2", O_WRONLY|O_CREAT|O_TRUNC,00777);
printf("x:%d\n", x);
int y;
close(1);
y=dup(x);
printf("x:%d y:%d\n", x, y);
write(1, "hi there", 8);
```

10) (Standard output redirection) Change the shell such that it can handle standard output redirection.

```
$ cat f1 > f3
```

will redirect the output of "cat f1" to file f3.