

운영체제 및 실습 - Task Programming -

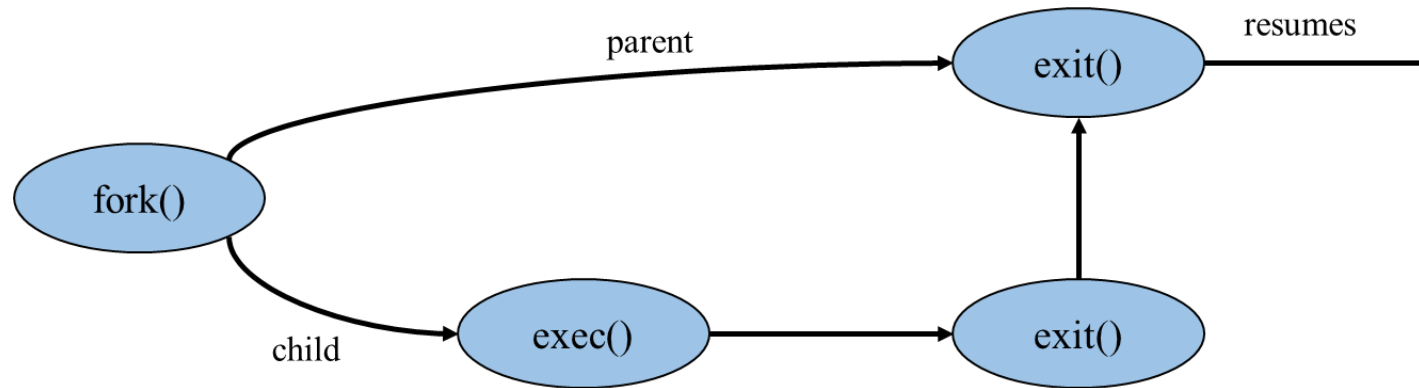
1

한예진, 김수창, 이승준

Dankook University

- Practice 1 : making two control flows
- Practice 2 : variable management
- Practice 3 : executing a new program
- Practice 4 : parameter passing to `main()` via shell
- Practice 5 : parameter passing to `main()` via `execle()`

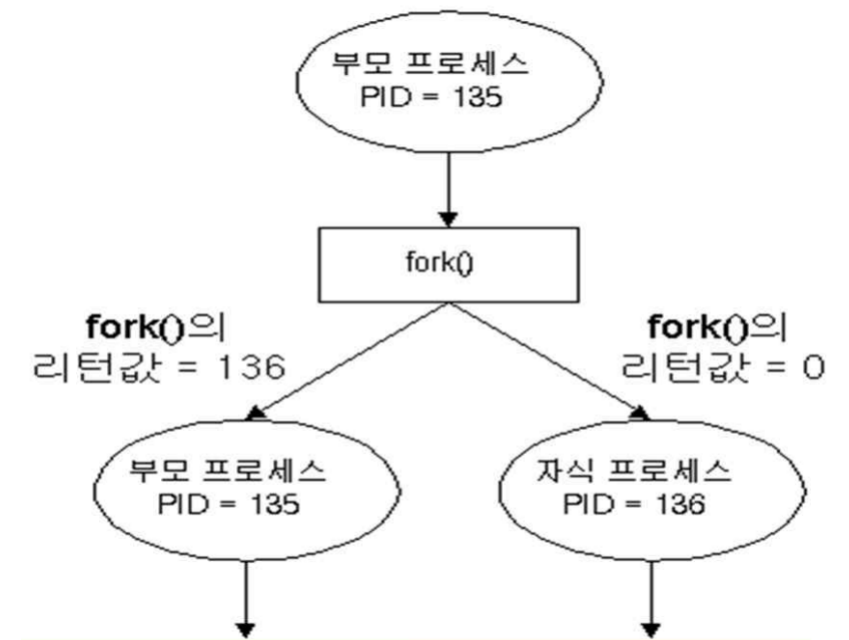
- Basic
 - `fork()`, `clone()` : create a task
 - `execve()` : execute a new program (binary loading)
 - `exit()` : terminate a task
 - `wait()`, `waitpid()` : wait for a task's termination (child or designated)
 - `getpid()` : get a task ID



Practice 1: fork()

4

- Make a new task whose memory image (text, data, ...) is the same as the existing task
 - Existing task: parent task
 - New task: child task
- Split the flow control into two (system's viewpoint)
 - One for parent and the other for child task
- Two return values (program's viewpoint)
 - Parent task: child's pid (always larger than 0)
 - Child task: 0
- wait()
 - wait for a task's termination (child or designated)

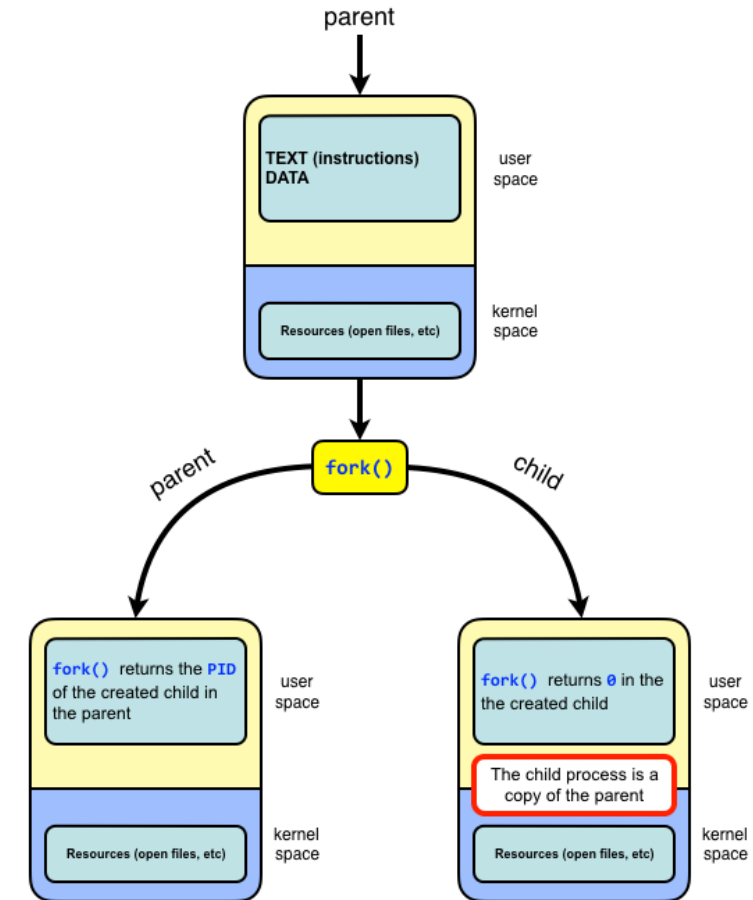


Practice 1: fork()

5

- fork_test.c
 - fork(), wait(), getpid(), getppid()

```
1 #include <sys/types.h>
2 #include <sys/wait.h>
3 #include <unistd.h>
4 #include <stdio.h>
5 #include <stdlib.h>
6
7 int main(){
8     pid_t fork_return;
9     int status;
10    printf("Hello, my pid is %d\n", getpid());
11
12    if((fork_return = fork()) < 0){
13        perror("fork error");
14        exit(1);
15    }
16    else if(fork_return == 0){    /* child process */
17        printf("child: pid = %d, ppid = %d\n", getpid(), getppid());
18    }
19    else{    /* parent process */
20
21        printf("parent: I created child with pid=%d\n", fork_return);
22    }
23
24    /* Following line is executed by both parent and child */
25    printf("Bye, my pid is %d\n", getpid());
26 }
27
28 /TABAFork_test.c
```



(Source: <https://www.it.uu.se/education/course/homepage/os/vt18/module-2/process-management/>)

Practice 1: fork()

- 실행파일

```
[ec2-user@ip-172-31-10-88 TABA]$ gcc -o fork_test fork_test.c
```

- 결과

```
[ec2-user@ip-172-31-30-110 task_programming]$ ./fork_test
Hello, my pid is 2082
child: pid = 2083, ppid = 2082
Bye, my pid is 2083
parent: I created child with pid=2083
Bye, my pid is 2082
```

Practice 2: fork()

7

- fork_test2.c
 - fork(), exit(), sleep(), getpid(), write()

```
1 #include <sys/types.h>
2 #include <sys/wait.h>
3 #include <unistd.h>
4 #include <stdio.h>
5 #include <stdlib.h>
6 int glob = 6;
7 char buf[] = "a write to stdout\n";
8
9 int main(){
10     int var = 88; pid_t fork_return;
11     if(write(STDOUT_FILENO, buf, sizeof(buf)) != sizeof(buf)){
12         perror("write error");
13         exit(1);
14     }
15     printf("before fork\n");          /* we don't flush stdout */
16     if((fork_return = fork()) < /* fill out here */){
17         perror("fork error");
18         exit(1);
19     }
20     else if(fork_return == /* fill out here */){ /* child process */
21         /* fill out here */ /* modify variables*/
22     }
23     else{ /* parent process */
24         /* fill out here */ /* scheduling */
25     }
26     printf("pid=%d, glob=%d, var=%d\n", getpid(), glob, var);
27
28     exit(0);
29 }
30
```

~/TABA/fork_test2.c~/TABA/fork_test2.c

```
[ec2-user@ip-172-31-10-88 TABA]$ ./fork_test2
a write to stdout
before fork
pid=62720, glob=7, var=89
pid=62719, glob=6, var=88
```

Practice 2: fork()

- 실행파일

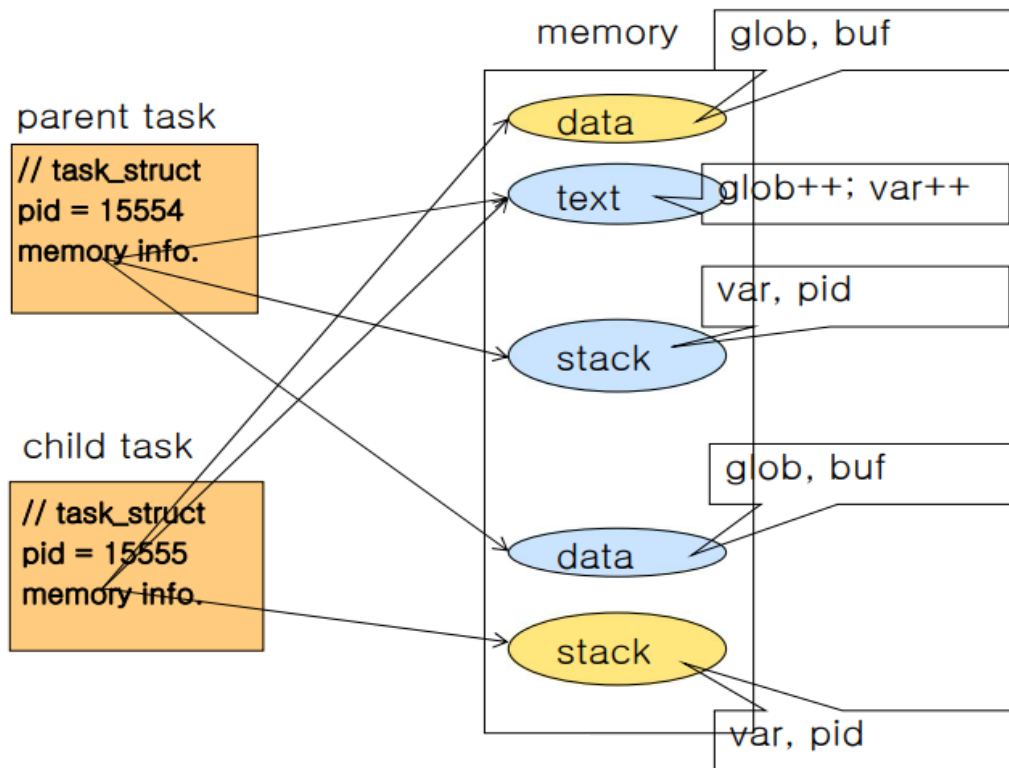
```
[ec2-user@ip-172-31-10-88 TABA]$ gcc -o fork_test2 fork_test2.c
```

- 결과

```
[ec2-user@ip-172-31-10-88 TABA]$ ./fork_test2  
a write to stdout  
before fork  
pid=62720, glob=7, var=89  
pid=62719, glob=6, var=88
```


Practice 2: fork()

- System's viewpoint of fork()

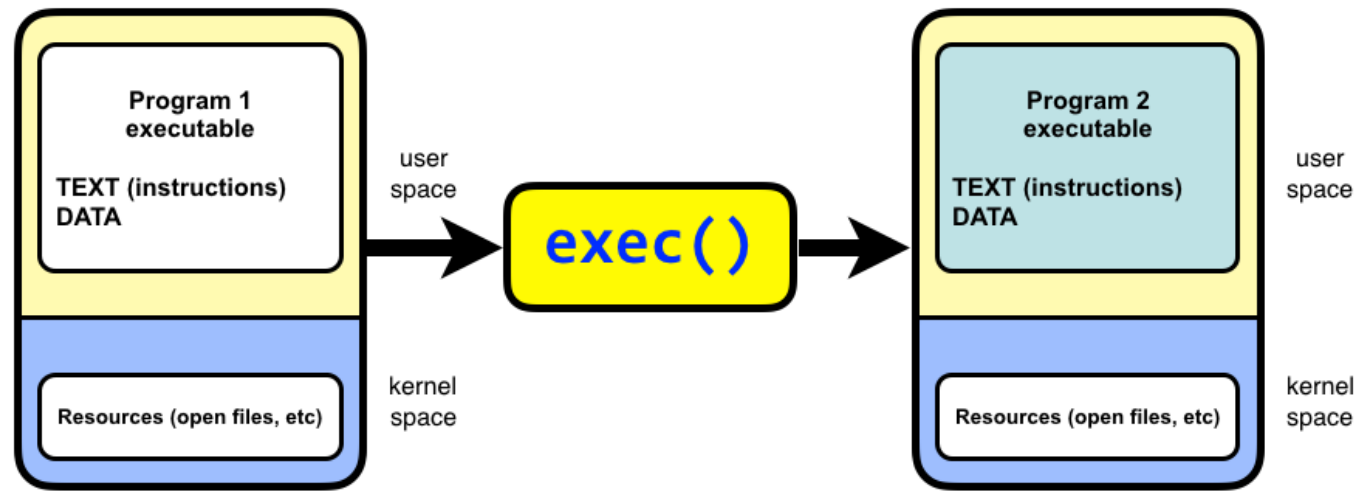


```
1 #include <sys/types.h>
2 #include <sys/wait.h>
3 #include <unistd.h>
4 #include <stdio.h>
5 #include <stdlib.h>
6 int glob = 6;
7 char buf[] = "a write to stdout\n";
8
9 int main(){
10     int var = 88; pid_t fork_return;
11     if(write(STDOUT_FILENO, buf, sizeof(buf)) != sizeof(buf)){
12         perror("write error");
13         exit(1);
14     }
15     printf("before fork\n");          /* we don't flush stdout */
16     if((fork_return = fork()) < 0){
17         perror("fork error");
18         exit(1);
19     }
20     else if(fork_return == 0){        /* child process */
21
22     }
23     else{                             /* parent process */
24         // wait(NULL);
25         sleep(2);
26     }
27     printf("pid=%d, glob=%d, var=%d\n", getpid(), glob, var);
28     exit(0);
29 }
30 }
```

Practice 3: `execve()`

10

- `execve()` system call
 - Execute a new program
 - Replace the current task's memory image (text, data, stack) with new binary
- System's viewpoint of `execve()`
 - Replace memory image (text, data, stack) with new one
 - The role of loader



(Source: <https://www.it.uu.se/education/course/homepage/os/vt18/module-2/exec/>)


Practice 3: execve()

11

- execl_test.c

- int execl(const char *filepath, const char *arg0, ... /* (char *) 0 */);

```
1 #include <sys/types.h>
2 #include <sys/wait.h>
3 #include <unistd.h>
4 #include <stdio.h>
5 #include <stdlib.h>
6
7 int main(){
8     pid_t fork_return, d_pid; int exit_status = -1;
9
10    if((fork_return = fork()) == -1){
11        // fork error handling
12        perror("fork error");
13        exit(1);
14    }
15    else if(fork_return == 0){ /* child process */
16        execl("./hello", "./hello", (char *)0);
17        printf("Child... I'm here\n");
18        // if execl() succeeds, the above printf() is not performed!!
19        exit(1);
20    }
21    else{ /* parent process */
22        d_pid = wait(&exit_status);
23        printf("Parent... I'm here\n");
24        printf("exit status of task %d is %d\n", d_pid, exit_status);
25    }
26 }
27
./TABA/execl_test.c
```



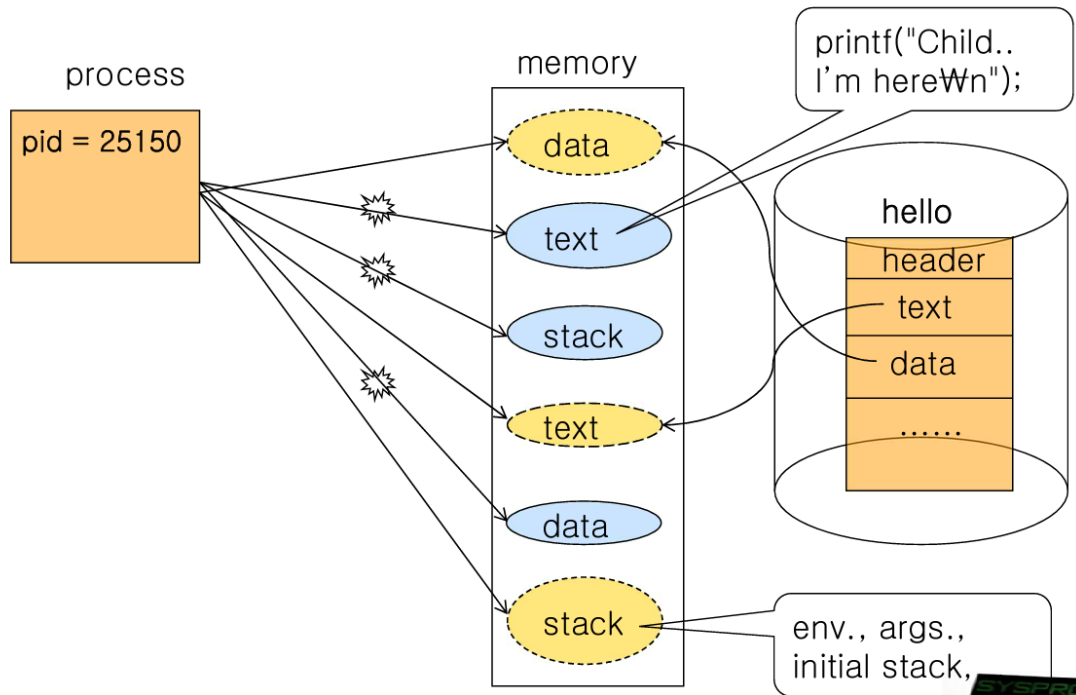
```
[ec2-user@ip-172-31-10-88 TABA]$ gcc -o hello hello.c
[ec2-user@ip-172-31-10-88 TABA]$ ./hello
Hello World
```

Practice 3: execve()

12

- execl_test.c

- int execl(const char *filepath, const char *arg0, ... /* (char *) 0 */);



```
1 #include <sys/types.h>
2 #include <sys/wait.h>
3 #include <unistd.h>
4 #include <stdio.h>
5 #include <stdlib.h>
6
7 int main(){
8     pid_t fork_return, d_pid; int exit_status = -1;
9
10    if((fork_return = fork()) == -1){
11        // fork error handling
12        perror("fork error");
13        exit(1);
14    }
15    else if(fork_return == 0){ /* child process */
16        execl("./hello", "./hello", (char *)0);
17        printf("Child... I'm here\\n");
18        // if execl() succeeds, the above printf() is not performed!!
19        exit(1);
20    }
21    else{ /* parent process */
22        d_pid = wait(&exit_status);
23        printf("Parent... I'm here\\n");
24        printf("exit status of task %d is %d\\n", d_pid, exit_status);
25    }
26 }
27
/TABA/execl_test.c
```

Practice 3: execve()

13

- 실행파일

```
[ec2-user@ip-172-31-10-88 TABA]$ gcc -o execl_test execl_test.c
```

- 결과

```
[ec2-user@ip-172-31-10-88 TABA]$ ./execl_test  
Hello World  
Parent... I'm here  
exit status of task 63792 is 0
```

Practice 4: parameter passing via shell

14

- execl_test2.c
 - Printing argv[] and env[]

```
1 #include <stdio.h>
2
3 int main(int argc, char * argv[], char *envp[]){
4     for(int i=0; argv[i]; i++)
5         printf("arg %d = %s\n", i, argv[i]);
6     for(int i=0; envp[i]; i++)
7         printf("env %d = %s\n", i, envp[i]);
8 }
9
```

~

```
~/task_programming/execl_test2.c/home/ec2-user/task_programming
```

Practice 4: parameter passing via shell

15

- 실행파일

```
[ec2-user@ip-172-31-10-88 TABA]$ gcc -o execl_test2 execl_test2.c
```

- 결과

```
[ec2-user@ip-172-31-10-88 TABA]$ ./execl_test2
arg 0 = ./execl_test2
env 0=SHELL=/bin/bash
env 1=HISTCONTROL=ignoredups
env 2=SYSTEMD_COLORS=false
env 3=HISTSIZE=1000
env 4=HOSTNAME=ip-172-31-10-88.ap-northeast-2.compute.internal
env 5=PWD=/home/ec2-user/TABA
env 6=LOGNAME=ec2-user
env 7=XDG_SESSION_TYPE=tty
env 8=MOTD_SHOWN=pam
env 9=HOME=/home/ec2-user
env 10=LANG=C.UTF-8
env 11=LS_COLORS=rs=0:di=01;34:ln=01;36:mh=00:pi=40;33:so=01;35:do=01;35:bd=40;33;01:cd=40;33;01:or=40;31;01:mi=01;37;41:su=3
7;41:sg=30;43:ca=30;41:tw=30;42:ow=34;42:st=37;44:ex=01;32:*.tar=01;31:*.tgz=01;31:*.arc=01;31:*.arj=01;31:*.taz=01;31:*.lha=
01;31:*.lz4=01;31:*.lzh=01;31:*.lzma=01;31:*.tlz=01;31:*.txz=01;31:*.tzo=01;31:*.t7z=01;31:*.zip=01;31:*.z=01;31:*.dz=01;31:*.
gz=01;31:*.lrz=01;31:*.lz=01;31:*.lzo=01;31:*.xz=01;31:*.zst=01;31:*.tzst=01;31:*.bz2=01;31:*.bz=01;31:*.tbz=01;31:*.tbz2=01
;31:*.tz=01;31:*.deb=01;31:*.rpm=01;31:*.jar=01;31:*.war=01;31:*.ear=01;31:*.sar=01;31:*.rar=01;31:*.alz=01;31:*.ace=01;31:*.
zoo=01;31:*.cpio=01;31:*.7z=01;31:*.rz=01;31:*.cab=01;31:*.wim=01;31:*.swm=01;31:*.dwm=01;31:*.esd=01;31:*.jpg=01;35:*.jpeg=0
1;35:*.mjpg=01;35:*.mjpeg=01;35:*.gif=01;35:*.bmp=01;35:*.pbm=01;35:*.pgm=01;35:*.ppm=01;35:*.tga=01;35:*.xbm=01;35:*.xpm=01;
35:*.tif=01;35:*.tiff=01;35:*.png=01;35:*.svg=01;35:*.svgz=01;35:*.mng=01;35:*.pcx=01;35:*.mov=01;35:*.mpg=01;35:*.mpeg=01;35
:*.m2v=01;35:*.mkv=01;35:*.webm=01;35:*.webp=01;35:*.ogm=01;35:*.mp4=01;35:*.m4v=01;35:*.mp4v=01;35:*.vob=01;35:*.qt=01;35:*.
nuv=01;35:*.wmv=01;35:*.asf=01;35:*.rm=01;35:*.rmvb=01;35:*.flc=01;35:*.avi=01;35:*.fli=01;35:*.flv=01;35:*.gl=01;35:*.dl=01;
35:*.xcf=01;35:*.xwd=01;35:*.yuv=01;35:*.cgm=01;35:*.emf=01;35:*.ogv=01;35:*.ogx=01;35:*.aac=01;36:*.au=01;36:*.flac=01;36:*.
m4a=01;36:*.mid=01;36:*.midi=01;36:*.mka=01;36:*.mp3=01;36:*.mpc=01;36:*.ogg=01;36:*.ra=01;36:*.wav=01;36:*.oga=01;36:*.opus=
01;36:*.spx=01;36:*.xspf=01;36:
```

Practice 5: parameter passing via execl()

16

- execl_test3.c

- `int execl(const char *filepath, const char *arg0, ... (char *) 0, char *const envp[]);`

```
1 #include<sys/types.h>
2 #include <sys/wait.h>
3 #include <unistd.h>
4 #include <stdio.h>
5 #include <stdlib.h>
6
7 int main(int argc, char *argv[]){
8     pid_t fork_return, d_pid; int exit_status;
9     char *const myenv[] = {"sys programing", "is", "fun", (char*)0};
10
11     if((fork_return = fork()) == -1){
12         // fork error handling
13         perror("fork error");
14         exit(1);
15     }
16     else if(fork_return == 0){ /* child process */
17         execl("./execl_test2", "./execl_test2", "Hi", "DKU", (char*)0, myenv);
18         printf("Child... I'm here\n");
19         // if execl() succeeds, the above printf() is not performed!!
20         exit(1);
21     }
22     else{ /* parent process */
23         d_pid = wait(&exit_status);
24         printf("exit pid = %d with status = %d\n", d_pid, WEXITSTATUS(exit_status));
25     }
26 }
27
~/TABA/execl_test3.c
```


Practice 5: parameter passing via execle()

17

- 실행파일

```
$ gcc -o execle_test3 execle_test3.c
```

- 결과

```
[ec2-user@ip-172-31-10-88 TABA]$ ./execle_test3  
arg 0 = ./execl_test2  
arg 1 = Hi  
arg 2 = DKU  
env 0 = sys programing  
env 1 = is  
env 2 = fun  
exit pid = 64616 with status = 0
```

Q&A