

## CSC 2405 – Running New Programs in Unix

### Exercise 1 – Running “ls” with “execl”

**execl.c**

```
#include <stdio.h>
#include <unistd.h>
#include <sys/types.h>

int main ()
{
    if(fork() == 0)
        execl("/bin/ls", "ls", "-l", 0);
    wait(NULL);
    printf("\nList completed. \n");
    return 0;
}
```

\*\*\*\*\*

Compile:                    gcc -o execl execl.c

Run:                        ./execl

\*\*\*\*\*

**Output:**

## Exercise 2 – Running “ls” with “execv”

**exec2.c**

```
#include <stdio.h>
#include <unistd.h>
#include <sys/types.h>

int main ()
{
    /* char * cmd[3] = {"/bin/ls", "-l", 0}; */

    char * cmd[3];

    cmd[0] = "/bin/ls";
    cmd[1] = "-l";
    cmd[2] = 0;
    if(fork() == 0)
        execl(cmd[0], cmd);
    wait(NULL);
    printf("\nList completed. \n");
    return 0;
}
```

```
*****
Compile:          gcc -o exec2 exec2.c
Run:              ./exec2
*****
```

**Output: Same.**

### Exercise 3 – Running “ls” with “execlp”

Try

```
echo $PATH
```

at the shell prompt. You will see a list of directories (search paths) separated by ':'. This is where the shell looks for a command you type in. Unlike `exec1` and `execv` that require the full path for a command, `execlp` and `execvp` search these directories for the command you specify as a first argument.

**exec3.c**

```
#include <stdio.h>
#include <unistd.h>
#include <sys/types.h>

int main ()
{
    if(fork() == 0)
        execlp("ls", "ls", "-l", 0);    /* not "/bin/ls" */
    wait(NULL);
    printf("\nList completed. \n");
    return 0;
}
```

```
*****
Compile:          gcc -o exec3 exec3.c
Run:              ./exec1
*****
```

**Output: Same.**

## Exercise 4 – Running Another Program

**exec4.c**

(executable **exec4**)

```
int main()
{
    pid = fork();
    if(pid == 0) {
        printf("Child runs\n");
        execl("atest", "atest", "dog", 0);
        printf("Child done\n");
    }
    else {
        wait(NULL);
        printf("Parent done\n");
    }
}
```

**atest.c**

(executable **atest**)

```
int main(int argc, char * argv[])
{
    printf("Got a [%s]", argv[1]);
}
```

a) If **execl** succeeds, what is the output of **exec4**?

b) If **execl** fails, what is the output of **exec4**?

c) Why would **execl** fail?

## Exercise 5 – Running “wc” on Multiple Files

The word count program “wc” can take as argument a file whose name is specified in the command line. For instance, the Unix command

```
wc -L filename
```

will print out the number of lines in the file with name `filename`.

Write a small program that takes any number of file names as command line parameters. For each filename, the program should create a child process, which

- a) prints out the name of the file
- b) uses `execlp` to execute the program “wc -L” on that file

The parent process should not exit, until all processes have exited.

**exec5.c**

```
int main(int argc, char * argv[])
{
    int i;
    for(i = 0; i < argc; i++)
    {
        if( fork() == 0)
        {
            printf("%s\n", argv[i]);
            execlp("wc", "wc", "-L", argv[i], 0);
            exit(1);
        }
    }
    for(i = 0; i < argc; i++)
        wait(NULL);
}
```