Adhesh Reghu Kumar
COE18B001

Operating Systems Lab

Lab Assignment - 3

(Only Test Drive Questions)

Note: The documentation of the test drive question for Lab 3 was missed out in the original documentation, so only the test drive questions have been included here.

Test Drive - 1: **Execl()**

Code

```c
/*****************************************
* AUTHOR : AdheshR *
* Problem Description: Test drive: execl*
*****************************************/
#include <stdio.h>
#include <stdlib.h>
#include <sys/types.h>
#include <unistd.h>
#include <inttypes.h>

int main()
{
    pid_t pid;
    pid = fork();
    if(pid == 0)
    {
       // Execute ls command
       int ret = execl("/bin/ls","ls",NULL);
       if (ret < 0)
             perror("Couldnt execute");
    }
    else
    {
       wait(NULL);
       printf("Parent has control\n");
       printf("Parent has waited for child to complete\n");
    }
    return 0;
}
```

Output

```
adheshreghu@adheshreghu-Inspiron-5570:~/Documents/SEM5/OS/Lab/Week3/1_test$ gcc -o out 1_execl.c
adheshreghu@adheshreghu-Inspiron-5570:~/Documents/SEM5/OS/Lab/Week3/1_test$ ./out
1_execl.c  1_wait.c  2_execlp.c  2_wait.c  3_execv.c  4_execvp.c  5_execve.c  out  output  try  try.c
Parent has control
Parent has waited for child to complete
adheshreghu@adheshreghu-Inspiron-5570:~/Documents/SEM5/OS/Lab/Week3/1_test$
```

Test Drive - 2: **Execlp()**

Code

```c
/*****************************************
* AUTHOR : AdheshR *
* Problem Description: Test drive: execlp*
*****************************************/
#include <stdio.h>
#include <stdlib.h>
#include <sys/types.h>
#include <unistd.h>
#include <inttypes.h>

int main()
{
    pid_t pid;
    pid = fork();

    if(pid == 0)
    {
       // Execute ls command
       int ret = execlp("ls","ls",NULL);
       if (ret < 0)
             perror("Couldnt execute");
    }
    else
    {
       wait(NULL);
       printf("Parent has control\n");
       printf("Parent has waited for child to complete\n");
    }
    return 0;
}
```

Output

```
adheshreghu@adheshreghu-Inspiron-5570:~/Documents/SEM5/OS/Lab/Week3/1_test$ gcc -o out 2_execlp.c
adheshreghu@adheshreghu-Inspiron-5570:~/Documents/SEM5/OS/Lab/Week3/1_test$ ./out
1_execl.c  1_wait.c  2_execlp.c  2_wait.c  3_execv.c  4_execvp.c  5_execve.c  out  output  try  try.c
Parent has control
Parent has waited for child to complete
adheshreghu@adheshreghu-Inspiron-5570:~/Documents/SEM5/OS/Lab/Week3/1_test$
```

Test Drive - 3: **Execv()**

Code

```c
/*****************************************
* AUTHOR : AdheshR *
* Problem Description: Test drive: execv*
*****************************************/
#include <stdio.h>
#include <stdlib.h>
#include <sys/types.h>
#include <unistd.h>
#include <inttypes.h>
char *const argv[] = {"ls",NULL};

int main()
{
    pid_t pid;
    pid = fork();
    if(pid == 0)
    {
       // Execute ls command
       int ret = execv("/bin/ls",argv);
       if (ret < 0)
             perror("Couldnt execute");
    }
    else
    {
       wait(NULL);
       printf("Parent has control\n");
       printf("Parent has waited for child to complete\n");
    }
    return 0;
}
```

Output

```
adheshreghu@adheshreghu-Inspiron-5570:~/Documents/SEM5/OS/Lab/Week3/1_test$ gcc -o out 3_execv.c
adheshreghu@adheshreghu-Inspiron-5570:~/Documents/SEM5/OS/Lab/Week3/1_test$ ./out
1_execl.c  1_wait.c  2_execlp.c  2_wait.c  3_execv.c  4_execvp.c  5_execve.c  out  output  try  try.c
Parent has control
Parent has waited for child to complete
adheshreghu@adheshreghu-Inspiron-5570:~/Documents/SEM5/OS/Lab/Week3/1_test$
```

Test Drive - 4: **Execvp()**

Code

```c
/*****************************************
* AUTHOR : AdheshR *
* Problem Description: Test drive: execvp*
*****************************************/
#include <stdio.h>
#include <stdlib.h>
#include <sys/types.h>
#include <unistd.h>
#include <inttypes.h>
char *const argv[] = {"ls","-l",NULL};

int main()
{
    pid_t pid;
    pid = fork();
    if(pid == 0)
    {
       // Execute ls command
       int ret = execvp(argv[0],argv);
       if (ret < 0)
             perror("Couldnt execute");
    }
    else
    {
       wait(NULL);
       printf("Parent has control\n");
       printf("Parent has waited for child to complete\n");
    }
    return 0;
}
```

Output

```
adheshreghu@adheshreghu-Inspiron-5570:~/Documents/SEM5/OS/Lab/Week3/1_test$ gcc -o out 4_execvp.c
adheshreghu@adheshreghu-Inspiron-5570:~/Documents/SEM5/OS/Lab/Week3/1_test$ ./out
total 60
-rw-rw-r-- 1 adheshreghu adheshreghu  554 Nov 29 12:57 1_execl.c
-rw-rw-r-- 1 adheshreghu adheshreghu  731 Nov 29 18:37 1_wait.c
-rw-rw-r-- 1 adheshreghu adheshreghu  551 Nov 29 12:59 2_execlp.c
-rw-rw-r-- 1 adheshreghu adheshreghu  726 Nov 29 18:42 2_wait.c
-rw-rw-r-- 1 adheshreghu adheshreghu  584 Nov 29 19:16 3_execv.c
-rw-rw-r-- 1 adheshreghu adheshreghu  589 Nov 29 19:20 4_execvp.c
-rw-rw-r-- 1 adheshreghu adheshreghu  618 Nov 29 17:30 5_execve.c
-rwxrwxr-x 1 adheshreghu adheshreghu 8624 Nov 29 19:21 out
drwxrwxr-x 2 adheshreghu adheshreghu 4096 Nov 29 19:20 output
-rwxrwxr-x 1 adheshreghu adheshreghu 8488 Nov 29 18:30 try
-rw-rw-r-- 1 adheshreghu adheshreghu  388 Nov 29 18:31 try.c
Parent has control
Parent has waited for child to complete
adheshreghu@adheshreghu-Inspiron-5570:~/Documents/SEM5/OS/Lab/Week3/1_test$
```

Test Drive - 5: **Execve()**

Code

```c
/*****************************************
* AUTHOR : AdheshR *
* Problem Description: Test drive: execve*
*****************************************/
#include <stdio.h>
#include <stdlib.h>
#include <sys/types.h>
#include <unistd.h>
#include <inttypes.h>

char *const argv[] = {"ls",NULL};
char *const env[] = {"HOME=/usr/home","LOGNAME=home",(char*)0};

int main()
{
    pid_t pid;
    pid = fork();

    if(pid == 0)
    {
       // Execute ls command
       int ret = execve("/bin/ls",argv,env);
```

```
        if (ret < 0)
              perror("Couldnt execute");
    }
    else
    {
        wait(NULL);
        printf("Parent has control\n");
        printf("Parent has waited for child to complete\n");
    }
    return 0;
}
```

Output

```
adheshreghu@adheshreghu-Inspiron-5570:~/Documents/SEM5/OS/Lab/Week3/1_test$ gcc -o out 5_execve.c
adheshreghu@adheshreghu-Inspiron-5570:~/Documents/SEM5/OS/Lab/Week3/1_test$ ./out
1_execl.c  1_wait.c  2_execlp.c  2_wait.c  3_execv.c  4_execvp.c  5_execve.c  out  output  try  try.c
Parent has control
Parent has waited for child to complete
adheshreghu@adheshreghu-Inspiron-5570:~/Documents/SEM5/OS/Lab/Week3/1_test$
```

Test Drive - 6: **Wait()**

Code

```
/*****************************************
* AUTHOR : AdheshR *
* Problem Description: Test drive: Wait()*
*****************************************/
#include <stdio.h>
#include <stdlib.h>
#include <sys/types.h>
#include <unistd.h>
#include <inttypes.h>
#define n 5

int main()
{
    pid_t child_pid,wpid;
    int status = 0;
```

```c
    // Parent code (before child processes starts)
    printf("Parent Process BEFORE children !!\n");

    int id = 0;
    for(id = 0; id < n; ++ id)
    {
       if((child_pid == fork()) == 0)
       {
             // Child code
             printf("Child - %d\n",id);
             exit(0);
       }
    }

    // Parent waits for all children
    while((wpid = wait(&status)) != -1);

    // Parent code (after all children)
    printf("Parent AFTER children !!\n");

    return 0;
}
```

Output

Test Drive - 7: **Waitpid()**

Code

```c
/*****************************************
* AUTHOR : AdheshR *
* Problem Description: Test drive: Wait()*
*****************************************/
#include <stdio.h>
#include <stdlib.h>
#include <sys/types.h>
#include <unistd.h>
#include <inttypes.h>

int main()
{
    pid_t child_pid;
    child_pid = fork();

    if(child_pid == 0)
    {
       // Do something
       printf("Inside child process.\n");
       exit(0);
    }
    else if(child_pid < 0)
       perror("Fork failure.\n");
    else
    {
       // Within parent
       int return_status;

       // Wait for the child
       waitpid(child_pid,&return_status,0);

       if(return_status == 0)
             printf("Child process terminated normally.\n");
       else
             printf("Child process terminated with an error.\n");

    }
    return 0;
}
```

Output

```
adheshreghu@adheshreghu-Inspiron-5570:~/Documents/SEM5/OS/Lab/Week3/1_test$ gcc -o out 2_wait.c
adheshreghu@adheshreghu-Inspiron-5570:~/Documents/SEM5/OS/Lab/Week3/1_test$ ./out
Inside child process.
Child process terminated normally.
adheshreghu@adheshreghu-Inspiron-5570:~/Documents/SEM5/OS/Lab/Week3/1_test$
```
