

# ***Operating Systems - EECE.5730***

***Instructor: Prof. Dalila Megherbi***

***Assignment – 1***

***Due by 09-22-16***

***By,***

***- Naga Ganesh Kurrapati***

## 1) Objective:

The objective of this assignment is to create parent and child processes using fork function and using other utility function to avoid abandoned or zombie child processes.

## 2) Background:

Fork () function creates the sub-process for the given process and operating system schedule those process to achieve the parallel processing. Using the wait () function, the parent process can be wait for its child processes to complete. The child notifies the parent by executing wait () function.

## 3) Algorithms/Functions used:

Fork () - To create sub-processes

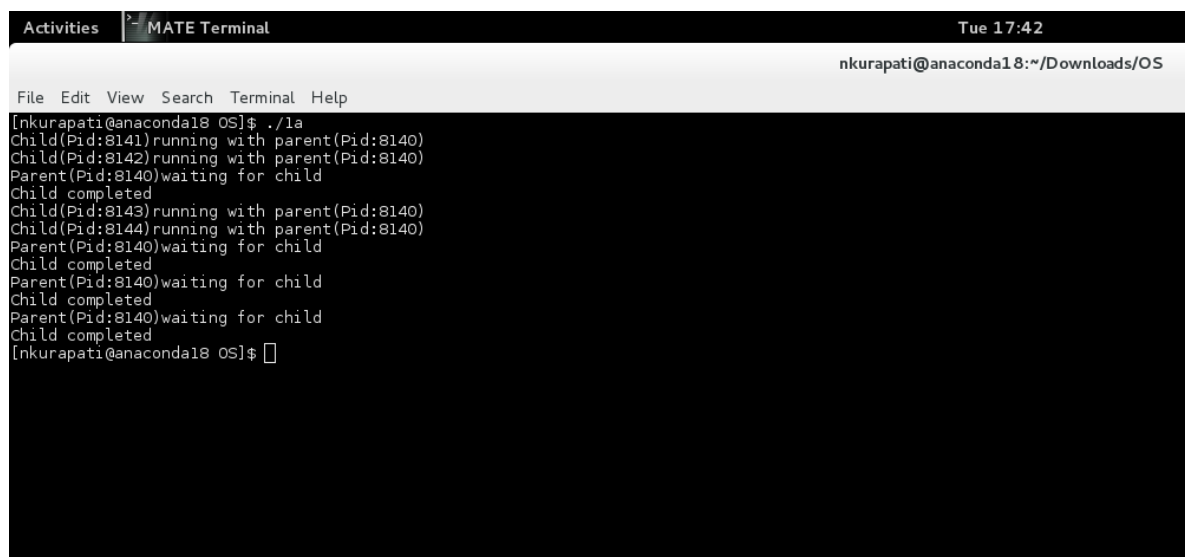
Wait () – to wait on a child process

Execve () – to spawn external programs

Exit () – to notify the parent of the job completion

Perror() – to notify the errors

## 4) Results:



```
Activities MATE Terminal Tue 17:42
nkurapati@anaconda18:~/Downloads/OS

File Edit View Search Terminal Help
[nkurapati@anaconda18 OS]$ ./1a
Child(Pid:8141)running with parent(Pid:8140)
Child(Pid:8142)running with parent(Pid:8140)
Parent(Pid:8140)waiting for child
Child completed
Child(Pid:8143)running with parent(Pid:8140)
Child(Pid:8144)running with parent(Pid:8140)
Parent(Pid:8140)waiting for child
Child completed
Parent(Pid:8140)waiting for child
Child completed
Parent(Pid:8140)waiting for child
Child completed
[nkurapati@anaconda18 OS]$
```

Output of program 1a

```
Activities MATE Terminal Tue 17:44
nkurapati@anaconda18:~/Downloads/OS

File Edit View Search Terminal Help

[nkurapati@anaconda18 OS]$ ./1b
Child(Pid:8341)running
Child(Pid:8342)running
Parent(Pid:8340)waiting for child
Child(Pid:8343)running
Child(Pid:8344)running
total 84
total 84
-rwxr-xr-x 1 nkurapati users 8907 Sep 20 17:36 1a
-rwxr-xr-x 1 nkurapati users 8907 Sep 20 17:36 1a
-rw-r--r-- 1 nkurapati users 755 Sep 20 17:36 1a.c
-rw-r--r-- 1 nkurapati users 755 Sep 20 17:36 1a.c
-rw-r--r-- 1 nkurapati users 752 Sep 18 23:57 1a.c~
-rw-r--r-- 1 nkurapati users 752 Sep 18 23:57 1a.c~
-rwxr-xr-x 1 nkurapati users 8905 Sep 20 17:36 1b
-rwxr-xr-x 1 nkurapati users 8905 Sep 20 17:36 1b
-rw-r--r-- 1 nkurapati users 918 Sep 18 23:39 1b.c
-rw-r--r-- 1 nkurapati users 918 Sep 18 23:39 1b.c
-rw-r--r-- 1 nkurapati users 918 Sep 18 16:31 1b.c~
-rw-r--r-- 1 nkurapati users 918 Sep 18 16:31 1b.c~
-rw-r--r-- 1 nkurapati users 618 Sep 18 10:06 1.c~
-rw-r--r-- 1 nkurapati users 618 Sep 18 10:06 1.c~
-rwxr-xr-x 1 nkurapati users 8906 Sep 20 17:36 2
-rwxr-xr-x 1 nkurapati users 8906 Sep 20 17:36 2
-rw-r--r-- 1 nkurapati users 1098 Sep 19 12:04 2.c
-rw-r--r-- 1 nkurapati users 1098 Sep 19 12:04 2.c
-rw-r--r-- 1 nkurapati users 1057 Sep 19 11:46 2.c~
-rw-r--r-- 1 nkurapati users 1057 Sep 19 11:46 2.c~
-rw-r--r-- 1 nkurapati users 113 Sep 19 17:09 Makefile
-rw-r--r-- 1 nkurapati users 113 Sep 19 17:09 Makefile
-rw-r--r-- 1 nkurapati users 116 Sep 19 17:07 Makefile~
-rw-r--r-- 1 nkurapati users 116 Sep 19 17:07 Makefile~
-rw-r--r-- 1 nkurapati users 1413 Sep 19 18:54 readme~
-rw-r--r-- 1 nkurapati users 1413 Sep 19 18:54 readme~
-rw-r--r-- 1 nkurapati users 1278 Sep 19 19:08 readme.txt
-rw-r--r-- 1 nkurapati users 1278 Sep 19 19:08 readme.txt
-rw-r--r-- 1 nkurapati users 1277 Sep 19 19:07 readme.txt~
-rw-r--r-- 1 nkurapati users 1277 Sep 19 19:07 readme.txt~
Child completed
Parent(Pid:8340)waiting for child
Child completed
total 84
Parent(Pid:8340)waiting for child
total 84
-rwxr-xr-x 1 nkurapati users 8907 Sep 20 17:36 1a
-rw-r--r-- 1 nkurapati users 755 Sep 20 17:36 1a.c
-rw-r--r-- 1 nkurapati users 752 Sep 18 23:57 1a.c~
-rwxr-xr-x 1 nkurapati users 8905 Sep 20 17:36 1b
-rw-r--r-- 1 nkurapati users 918 Sep 18 23:39 1b.c
-rw-r--r-- 1 nkurapati users 918 Sep 18 16:31 1b.c~
-rwxr-xr-x 1 nkurapati users 8907 Sep 20 17:36 1a
-rw-r--r-- 1 nkurapati users 618 Sep 18 10:06 1.c~
-rwxr-xr-x 1 nkurapati users 8906 Sep 20 17:36 2
-rw-r--r-- 1 nkurapati users 1098 Sep 19 12:04 2.c
-rw-r--r-- 1 nkurapati users 755 Sep 20 17:36 1a.c
-rw-r--r-- 1 nkurapati users 1057 Sep 19 11:46 2.c~
-rw-r--r-- 1 nkurapati users 113 Sep 19 17:09 Makefile
-rw-r--r-- 1 nkurapati users 752 Sep 18 23:57 1a.c~
-rw-r--r-- 1 nkurapati users 116 Sep 19 17:07 Makefile~
-rw-r--r-- 1 nkurapati users 1413 Sep 19 18:54 readme~
-rwxr-xr-x 1 nkurapati users 8905 Sep 20 17:36 1b
-rw-r--r-- 1 nkurapati users 1278 Sep 19 19:08 readme.txt
-rw-r--r-- 1 nkurapati users 918 Sep 18 23:39 1b.c
-rw-r--r-- 1 nkurapati users 1277 Sep 19 19:07 readme.txt~
```

Output of program 1b cont.

```
Activities MATE Terminal Tue 17:44
nkurapati@anaconda18:~/Downloads/OS

File Edit View Search Terminal Help
-rwxr-xr-x 1 nkurapati users 8905 Sep 20 17:36 1b
-rwxr-xr-x 1 nkurapati users 8905 Sep 20 17:36 1b
-rw-r--r-- 1 nkurapati users 918 Sep 18 23:39 1b.c
-rw-r--r-- 1 nkurapati users 918 Sep 18 23:39 1b.c
-rw-r--r-- 1 nkurapati users 918 Sep 18 16:31 1b.c~
-rw-r--r-- 1 nkurapati users 918 Sep 18 16:31 1b.c~
-rw-r--r-- 1 nkurapati users 618 Sep 18 10:06 1.c~
-rw-r--r-- 1 nkurapati users 618 Sep 18 10:06 1.c~
-rwxr-xr-x 1 nkurapati users 8906 Sep 20 17:36 2
-rwxr-xr-x 1 nkurapati users 8906 Sep 20 17:36 2
-rw-r--r-- 1 nkurapati users 1098 Sep 19 12:04 2.c
-rw-r--r-- 1 nkurapati users 1098 Sep 19 12:04 2.c
-rw-r--r-- 1 nkurapati users 1057 Sep 19 11:46 2.c~
-rw-r--r-- 1 nkurapati users 1057 Sep 19 11:46 2.c~
-rw-r--r-- 1 nkurapati users 113 Sep 19 17:09 Makefile
-rw-r--r-- 1 nkurapati users 113 Sep 19 17:09 Makefile
-rw-r--r-- 1 nkurapati users 116 Sep 19 17:07 Makefile~
-rw-r--r-- 1 nkurapati users 116 Sep 19 17:07 Makefile~
-rw-r--r-- 1 nkurapati users 1413 Sep 19 18:54 readme~
-rw-r--r-- 1 nkurapati users 1413 Sep 19 18:54 readme~
-rw-r--r-- 1 nkurapati users 1278 Sep 19 19:08 readme.txt
-rw-r--r-- 1 nkurapati users 1278 Sep 19 19:08 readme.txt
-rw-r--r-- 1 nkurapati users 1277 Sep 19 19:07 readme.txt~
-rw-r--r-- 1 nkurapati users 1277 Sep 19 19:07 readme.txt~
Child completed
Parent(Pid:8340)waiting for child
Child completed
total 84
Parent(Pid:8340)waiting for child
total 84
-rwxr-xr-x 1 nkurapati users 8907 Sep 20 17:36 1a
-rw-r--r-- 1 nkurapati users 755 Sep 20 17:36 1a.c
-rw-r--r-- 1 nkurapati users 752 Sep 18 23:57 1a.c~
-rwxr-xr-x 1 nkurapati users 8905 Sep 20 17:36 1b
-rw-r--r-- 1 nkurapati users 918 Sep 18 23:39 1b.c
-rw-r--r-- 1 nkurapati users 918 Sep 18 16:31 1b.c~
-rwxr-xr-x 1 nkurapati users 8907 Sep 20 17:36 1a
-rw-r--r-- 1 nkurapati users 618 Sep 18 10:06 1.c~
-rwxr-xr-x 1 nkurapati users 8906 Sep 20 17:36 2
-rw-r--r-- 1 nkurapati users 1098 Sep 19 12:04 2.c
-rw-r--r-- 1 nkurapati users 755 Sep 20 17:36 1a.c
-rw-r--r-- 1 nkurapati users 1057 Sep 19 11:46 2.c~
-rw-r--r-- 1 nkurapati users 113 Sep 19 17:09 Makefile
-rw-r--r-- 1 nkurapati users 752 Sep 18 23:57 1a.c~
-rw-r--r-- 1 nkurapati users 116 Sep 19 17:07 Makefile~
-rw-r--r-- 1 nkurapati users 1413 Sep 19 18:54 readme~
-rwxr-xr-x 1 nkurapati users 8905 Sep 20 17:36 1b
-rw-r--r-- 1 nkurapati users 1278 Sep 19 19:08 readme.txt
-rw-r--r-- 1 nkurapati users 918 Sep 18 23:39 1b.c
-rw-r--r-- 1 nkurapati users 1277 Sep 19 19:07 readme.txt~
-rw-r--r-- 1 nkurapati users 918 Sep 18 16:31 1b.c~
-rw-r--r-- 1 nkurapati users 618 Sep 18 10:06 1.c~
-rwxr-xr-x 1 nkurapati users 8906 Sep 20 17:36 2
-rw-r--r-- 1 nkurapati users 1098 Sep 19 12:04 2.c
-rw-r--r-- 1 nkurapati users 1057 Sep 19 11:46 2.c~
-rw-r--r-- 1 nkurapati users 113 Sep 19 17:09 Makefile
-rw-r--r-- 1 nkurapati users 116 Sep 19 17:07 Makefile~
Child completed
Parent(Pid:8340)waiting for child
-rw-r--r-- 1 nkurapati users 1413 Sep 19 18:54 readme~
-rw-r--r-- 1 nkurapati users 1278 Sep 19 19:08 readme.txt
-rw-r--r-- 1 nkurapati users 1277 Sep 19 19:07 readme.txt~
Child completed
[nkurapati@anaconda18 OS]$
```

Output of program 1b

```
Activities MATE Terminal Tue 17:45
nkurapati@anaconda18:~/Downloads/OS
File Edit View Search Terminal Help
[nkurapati@anaconda18 OS]$ ./2
Segmentation fault (core dumped)
[nkurapati@anaconda18 OS]$ ./2 8
I am process 1, My Pid:8424 and My parent Pid:8423
I am process 2, My Pid:8425 and My parent Pid:8424
I am process 3, My Pid:8426 and My parent Pid:8425
I am process 4, My Pid:8427 and My parent Pid:8426
I am process 5, My Pid:8428 and My parent Pid:8427
I am process 6, My Pid:8429 and My parent Pid:8428
I am process 7, My Pid:8430 and My parent Pid:8429
I am process 8, My Pid:8431 and My parent Pid:8430
I am process 9, My Pid:8432 and My parent Pid:8431
I am process 10, My Pid:8433 and My parent Pid:8432
I am process 11, My Pid:8434 and My parent Pid:8433
I am process 12, My Pid:8435 and My parent Pid:8434
I am process 13, My Pid:8436 and My parent Pid:8435
I am process 14, My Pid:8437 and My parent Pid:8436
I am process 15, My Pid:8438 and My parent Pid:8437
I am process 16, My Pid:8439 and My parent Pid:8438
[nkurapati@anaconda18 OS]$
```

Output of program 2

## 5) Observations:

Each process has unique process id. The fork returns 0 for the child process created and the child Pid to the parent.

## 6) Conclusions:

Successfully created the parent and child processes according to the problem designed. The job assigned accordingly and parent wait for the its child to terminate.

## 7) Source Code:

```

/*-----Program 1a.c-----*/
#include<sys/types.h>
#include<sys/wait.h>
#include<stdio.h>
#include<unistd.h>
#include<stdlib.h>
int main()
{
    unsigned int childNo = 4;
    pid_t pid[childNo];
    unsigned int i;
    for(i=0;i<childNo;i++)
    {
        /* fork a child process */
        pid[i] = fork();
        /* fork fails */
        if (pid[i] < 0)
        {
            perror("fork");
            return 1;
        }
        /* child process */
        if (pid[i] == 0)
        {
            printf("Child(Pid:%d)running with parent(Pid:%d)\n",getpid(),getppid());
            sleep(5);
            exit(0);
        }
        /* parent process */
        for(i=0;i<childNo;i++)
        {
            printf("Parent(Pid:%d)waiting for child\n",getpid());
            /* wait for the child to complete its job*/
            wait(NULL);
            printf("Child completed\n");
        }
        return 0;
    }
}

```

```

/*-----Program 1b.c-----*/
#include<sys/types.h>
#include<sys/wait.h>
#include<stdio.h>
#include<unistd.h>
#include<stdlib.h>
int main()
{
    unsigned int childNo = 4;
    pid_t pid[childNo];
    unsigned int i;
    for(i=0;i<childNo;i++)
    {
        /* fork a child process */
        pid[i] = fork();
        /* fork fails */
        if (pid[i] < 0)
        {
            perror("fork");
            exit(1);
        }
        /* child process */
        if (pid[i] == 0)
        {
            printf("Child(Pid:%d)running\n",getpid());
            char *args[] = {NULL};
            int res = execv("./b", args);
            /* return from execv if error occurs*/
            if (res == -1)
            {
                perror("execv");
                exit(2);
            }
            exit(0);
        }
        /* parent process */
        for(i=0;i<childNo;i++)
        {
            printf("Parent(Pid:%d)waiting for child\n",getpid());
            /* wait for the child to complete its job*/
            wait(NULL);
            printf("Child completed\n");
        }
    }
    return 0;
}

```

```
/*-----Program b.c-----*/
#include<sys/types.h>
#include<sys/wait.h>
#include<stdio.h>
#include<unistd.h>
#include<stdlib.h>
int main()
{
char *args[] = {"ls","-l",NULL};
int tmp = execv("/bin/ls",args);
if(tmp==-1)
{
perror("execv");
exit(2);
}
return 0;
}
```



```

/*-----Program 2.c-----*/
#include<sys/types.h>
#include<sys/wait.h>
#include<stdio.h>
#include<unistd.h>
#include<stdlib.h>
int main(int argc, char *argv[])
{
    unsigned int processesNo = atoi(argv[1]);
    pid_t pid;
    unsigned int processNo=0;
    unsigned int childNo=0;
    processNo++;
    /* fork a child process */
    pid = fork();
    /* fork fails */
    if (pid < 0)
    {
        perror("fork");
        exit(1);
    }
    /* child process */
    if (pid == 0)
    {
        while(childNo < 1 )
        {
            if(childNo == 0)
            printf("I am process %d, My Pid:%d and My parent
            Pid:%d\n",processNo,getpid(),getppid());
            processNo++;
            pid = fork(); /* fork a child process */
            /* fork fails */
            if (pid < 0)
            {
                perror("fork");
                exit(1);
            }
            if(pid == 0)
            {
                childNo = 0;
                if(processNo == (2*processesNo+1))
                exit(0);
            }
            else
            {
                childNo++;
            }
        }
    }
}

```

```
/* wait for the child to complete its job*/  
wait(NULL);  
exit(0);  
}  
}  
}  
/* parent process */  
else  
{  
/* wait for the child to complete its job*/  
wait(NULL);  
}  
return 0;  
}
```