

Subject : Operating Systems
Lab 7 Assignment
Date : 30/08/2020
Name : Devang Bhalala
Roll no: 007

Aim : Study of Dup2 system call and Exec1 function.

Theory Explanation:

1) dup2 system call:

SYNOPSIS

```
#include <unistd.h>
int dup2(int oldfd, int newfd);
```

DESCRIPTION

This system call creates a copy of the file descriptor oldfd.

dup2() makes newfd be the copy of oldfd, closing newfd first if necessary, but note the following:

- * If oldfd is not a valid file descriptor, then the call fails, and newfd is not closed.

- * If oldfd is a valid file descriptor, and newfd has the same value as oldfd, then dup2() does nothing, and returns newfd.

After a successful return from this system call, the old and new file descriptors may be used interchangeably. They refer to the same open file description and thus share file offset and file status flags.

The two descriptors do not share file descriptor flags (the close-on-exec flag). The close-on-exec flag for the duplicate descriptor is off.

RETURN VALUE

On success, this system call returns the new descriptor. On error, -1 is returned, and errno is set appropriately.

2) execl function:

SYNOPSIS

```
#include <unistd.h>
extern char **environ;
int execl(const char *path, const char *arg, ...);
```

DESCRIPTION

The const char *arg and subsequent ellipses in the execl(), execlp(), and execlx() functions can be thought of as arg0, arg1, ..., argn. Together they describe a list of one or more pointers to null-terminated strings that represent the argument list available to the executed program.

The first argument, by convention, should point to the filename associated with the file being executed.

The list of arguments must be terminated by a NULL pointer, and, since these are variadic functions, this pointer must be cast (char *) NULL.

RETURN VALUE

The execl() functions only return if an error has occurred. The return value is -1, and errno is set to indicate the error.

ASSIGNMENT PROGRAMS:

PROGRAM 1:

Write a program to implement ls | sort functionality using the system calls and functions covered in the lab.

CODE:

```
#include<stdio.h>
#include<unistd.h>
#include<sys/wait.h>
#include<sys/stat.h>
#include<fcntl.h>
#include<string.h>

void main()
{
    int pid,pret,pipefd[2],wst;
    pret = pipe(pipefd);
    if (pret == -1)
    {
        printf("\n\n--Task failed. Could not create pipe.--\n\n");
    }
}
```

```

else
{
    pid = fork();
    if (pid == -1)
    {
        printf("\n\n--Task failed. Could not create pipe.--\n\n");
    }
    else if (pid > 0)
    {
        wait(&wst);
        close(pipefd[1]);
        printf("---Entered in Parent process---\n\n");
        dup2(pipefd[0], 0);
        execl("/bin/sort", "sort", (char*)NULL);
    }
    else{
        close(pipefd[0]);
        printf("---Entered in Child process---\n\n");
        dup2(pipefd[1], 1);
        execl("/bin/ls", "ls", "-l", (char*)NULL);
    }
}
}
}

```

OUTPUT:

```

devangbhalala@devangbhalala-VirtualBox: ~/OS_labwork/lab7
devangbhalala@devangbhalala-VirtualBox:~/OS_labwork/lab7$ ls -l
total 44
-rwxrwxr-x 1 devangbhalala devangbhalala 17168 Aug 30 17:32 a.out
-rw-rw-r-- 1 devangbhalala devangbhalala 709 Aug 30 17:42 prog1.c
-rw-rw-r-- 1 devangbhalala devangbhalala 1091 Aug 30 17:43 prog2.c
-rw-rw-r-- 1 devangbhalala devangbhalala 235 Aug 30 16:13 task1.c
-rw-rw-r-- 1 devangbhalala devangbhalala 173 Aug 30 16:22 task2_1.c
-rw-rw-r-- 1 devangbhalala devangbhalala 417 Aug 30 16:29 task2_2.c
-rw-rw-r-- 1 devangbhalala devangbhalala 44 Aug 30 17:34 Text.txt
devangbhalala@devangbhalala-VirtualBox:~/OS_labwork/lab7$
devangbhalala@devangbhalala-VirtualBox:~/OS_labwork/lab7$ gcc prog1.c
devangbhalala@devangbhalala-VirtualBox:~/OS_labwork/lab7$ ./a.out
---Entered in Child process---
---Entered in Parent process---
-rw-rw-r-- 1 devangbhalala devangbhalala 1091 Aug 30 17:43 prog2.c
-rw-rw-r-- 1 devangbhalala devangbhalala 173 Aug 30 16:22 task2_1.c
-rw-rw-r-- 1 devangbhalala devangbhalala 235 Aug 30 16:13 task1.c
-rw-rw-r-- 1 devangbhalala devangbhalala 417 Aug 30 16:29 task2_2.c
-rw-rw-r-- 1 devangbhalala devangbhalala 44 Aug 30 17:34 Text.txt
-rw-rw-r-- 1 devangbhalala devangbhalala 709 Aug 30 17:42 prog1.c
-rwxrwxr-x 1 devangbhalala devangbhalala 16992 Aug 30 17:52 a.out
total 44
devangbhalala@devangbhalala-VirtualBox:~/OS_labwork/lab7$

```

```

devangbhalala@devangbhalala-VirtualBox: ~
devangbhalala@devangbhalala-VirtualBox:~$ ls -l
total 56
-rwxrwxr-x 1 devangbhalala devangbhalala 16992 Aug 30 17:52 a.out
drwxr-xr-x 2 devangbhalala devangbhalala 4096 Jul 13 19:19 Desktop
drwxr-xr-x 2 devangbhalala devangbhalala 4096 Jul 13 19:19 Documents
drwxr-xr-x 2 devangbhalala devangbhalala 4096 Aug 11 15:18 Downloads
drwxr-xr-x 2 devangbhalala devangbhalala 4096 Jul 13 19:19 Music
drwxrwxr-x 9 devangbhalala devangbhalala 4096 Aug 30 15:28 OS_labwork
drwxr-xr-x 2 devangbhalala devangbhalala 4096 Jul 13 19:19 Pictures
drwxr-xr-x 2 devangbhalala devangbhalala 4096 Jul 13 19:19 Public
drwxr-xr-x 2 devangbhalala devangbhalala 4096 Jul 13 19:19 Templates
drwxr-xr-x 2 devangbhalala devangbhalala 4096 Jul 13 19:19 Videos
devangbhalala@devangbhalala-VirtualBox:~$
devangbhalala@devangbhalala-VirtualBox:~$ ./a.out
---Entered in Child process---
---Entered in Parent process---

drwxrwxr-x 9 devangbhalala devangbhalala 4096 Aug 30 15:28 OS_labwork
drwxr-xr-x 2 devangbhalala devangbhalala 4096 Aug 11 15:18 Downloads
drwxr-xr-x 2 devangbhalala devangbhalala 4096 Jul 13 19:19 Desktop
drwxr-xr-x 2 devangbhalala devangbhalala 4096 Jul 13 19:19 Documents
drwxr-xr-x 2 devangbhalala devangbhalala 4096 Jul 13 19:19 Music
drwxr-xr-x 2 devangbhalala devangbhalala 4096 Jul 13 19:19 Pictures
drwxr-xr-x 2 devangbhalala devangbhalala 4096 Jul 13 19:19 Public
drwxr-xr-x 2 devangbhalala devangbhalala 4096 Jul 13 19:19 Templates
drwxr-xr-x 2 devangbhalala devangbhalala 4096 Jul 13 19:19 Videos
-rwxrwxr-x 1 devangbhalala devangbhalala 16992 Aug 30 17:52 a.out
total 56
devangbhalala@devangbhalala-VirtualBox:~$ █

devangbhalala@devangbhalala-VirtualBox: ~/OS_labwork/lab7/EmptyFolder
devangbhalala@devangbhalala-VirtualBox:~/OS_labwork/lab7/EmptyFolder$ ls -l
total 20
-rwxrwxr-x 1 devangbhalala devangbhalala 16992 Aug 30 17:52 a.out
devangbhalala@devangbhalala-VirtualBox:~/OS_labwork/lab7/EmptyFolder$ ./a.out
---Entered in Child process---
---Entered in Parent process---

-rwxrwxr-x 1 devangbhalala devangbhalala 16992 Aug 30 17:52 a.out
total 20
devangbhalala@devangbhalala-VirtualBox:~/OS_labwork/lab7/EmptyFolder$ █

```

PROGRAM 2:

Write a program to achieve following:

=> Child process should open a file with the contents to be sorted,
pass the contents to parent process.

=> Parent process should sort the contents of the file and display.

CODE:

```
#include<stdio.h>
#include<unistd.h>
#include<sys/wait.h>
#include<sys/stat.h>
#include<fcntl.h>
#include<string.h>

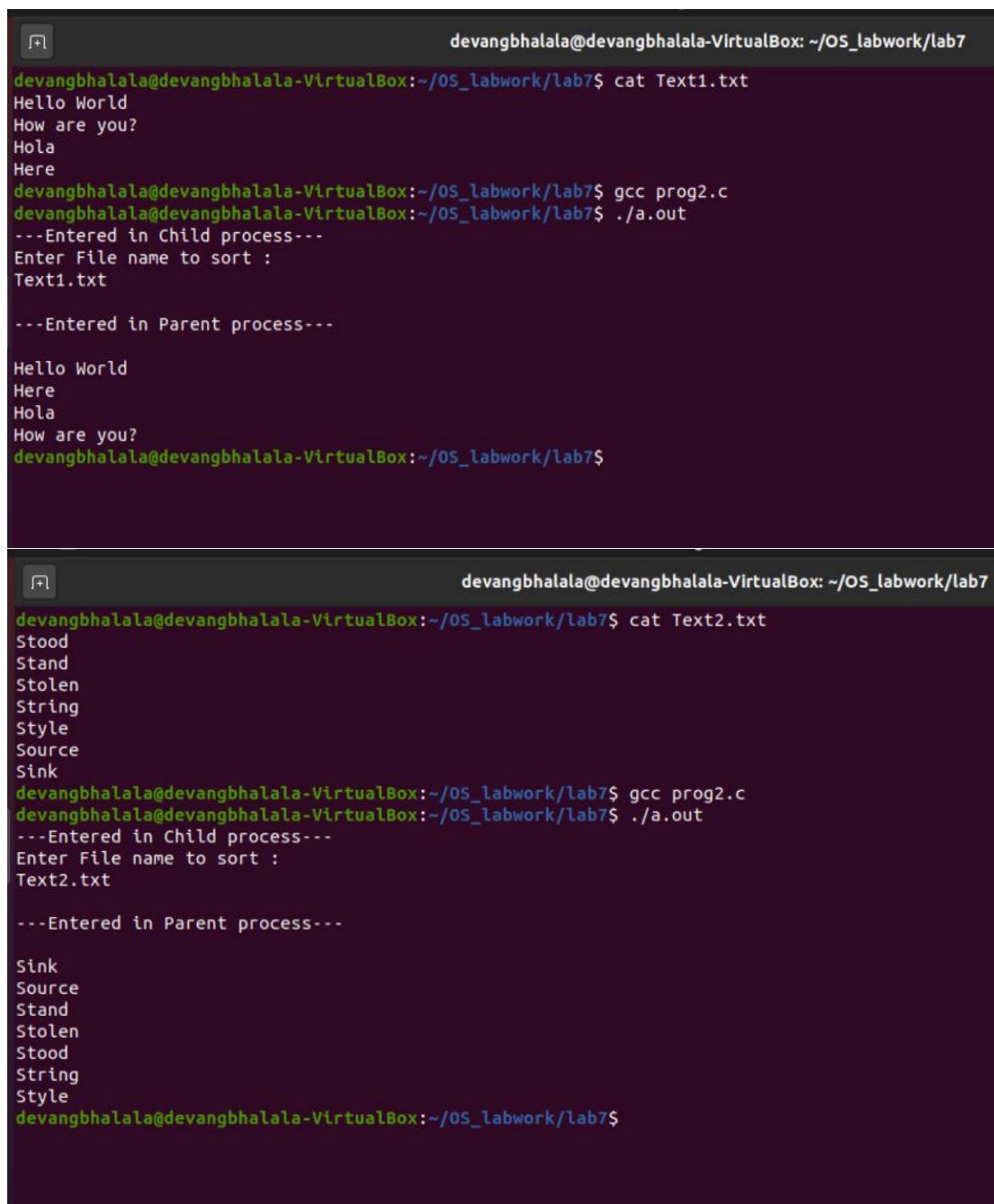
void main()
{
    int pid,pret,pipefd[2],wst,filehand,rd;
    char filedata[1000];
    char filenm[100];
    pret = pipe(pipefd);
    if (pret == -1)
    {
        printf("\n\n--Task failed. Could not create pipe.--\n\n");
    }
    else
    {
        pid = fork();
        if (pid == -1)
        {
            printf("\n\n--Task failed. Could not create pipe.--\n\n");
;
        }
        else if (pid > 0)
        {
            wait(&wst);
            close(pipefd[1]);
            printf("\n---Entered in Parent process---\n\n");
            dup2(pipefd[0], 0);
            execl("/bin/sort","sort",(char*)NULL);
        }
        else{
            close(pipefd[0]);
            printf("---Entered in Child process---\n\n");
            printf("Enter File name to sort : \n");
            bzero(filenm, sizeof(filenm));
```

```

        rd = read(0, filenm, sizeof(filenm));
        filenm[strlen(filenm) - 1] = '\0';
        filehand = open(filenm, O_RDONLY);
        bzero(filedata, sizeof(filedata));
        rd = read(filehand, filedata, sizeof(filedata));
        write(pipefd[1], filedata, rd);
    }
}
}

```

OUTPUT:



The image shows two screenshots of a terminal window. The first screenshot shows the execution of a program that sorts the contents of 'Text1.txt'. The second screenshot shows the execution of the same program with 'Text2.txt'.

```

devangbhalala@devangbhalala-VirtualBox: ~/OS_labwork/lab7
devangbhalala@devangbhalala-VirtualBox:~/OS_labwork/lab7$ cat Text1.txt
Hello World
How are you?
Hola
Here
devangbhalala@devangbhalala-VirtualBox:~/OS_labwork/lab7$ gcc prog2.c
devangbhalala@devangbhalala-VirtualBox:~/OS_labwork/lab7$ ./a.out
---Entered in Child process---
Enter File name to sort :
Text1.txt

---Entered in Parent process---

Hello World
Here
Hola
How are you?
devangbhalala@devangbhalala-VirtualBox:~/OS_labwork/lab7$

```

```

devangbhalala@devangbhalala-VirtualBox: ~/OS_labwork/lab7
devangbhalala@devangbhalala-VirtualBox:~/OS_labwork/lab7$ cat Text2.txt
Stood
Stand
Stolen
String
Style
Source
Sink
devangbhalala@devangbhalala-VirtualBox:~/OS_labwork/lab7$ gcc prog2.c
devangbhalala@devangbhalala-VirtualBox:~/OS_labwork/lab7$ ./a.out
---Entered in Child process---
Enter File name to sort :
Text2.txt

---Entered in Parent process---

Sink
Source
Stand
Stolen
Stood
String
Style
devangbhalala@devangbhalala-VirtualBox:~/OS_labwork/lab7$

```