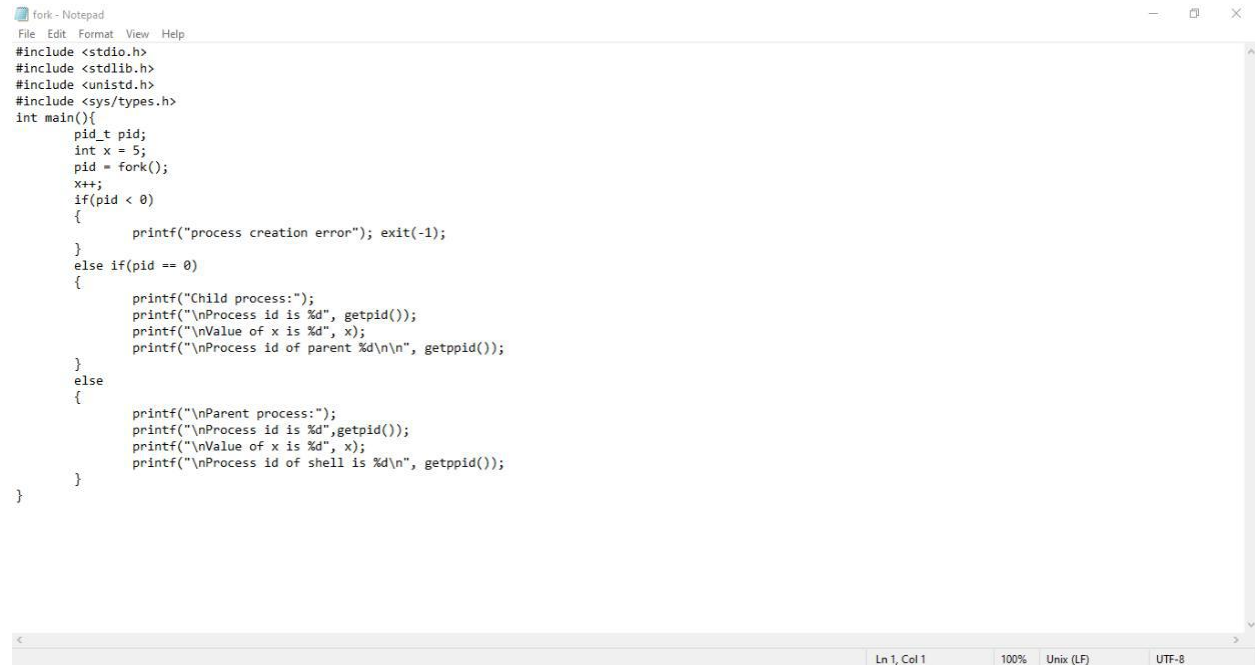NAMA : Auzan Danar Kusuma
NIM   : L200180005
KELAS : A

Praktikum Modul ke 8

Fork.c

```
fork - Notepad
File  Edit  Format  View  Help
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <sys/types.h>
int main(){
        pid_t pid;
        int x = 5;
        pid = fork();
        x++;
        if(pid < 0)
        {
                printf("process creation error"); exit(-1);
        }
        else if(pid == 0)
        {
                printf("Child process:");
                printf("\nProcess id is %d", getpid());
                printf("\nValue of x is %d", x);
                printf("\nProcess id of parent %d\n\n", getppid());
        }
        else
        {
                printf("\nParent process:");
                printf("\nProcess id is %d",getpid());
                printf("\nValue of x is %d", x);
                printf("\nProcess id of shell is %d\n", getppid());
        }
}
```

Ln 1, Col 1   100%   Unix (LF)   UTF-8

```
Terminal - kayon-pc11@kayon-pc11: ~

File   Edit   View   Terminal   Tabs   Help

kayon-pc11@kayon-pc11:~$ gcc fork.c
kayon-pc11@kayon-pc11:~$ ./a.out

Parent process:
Process id is 20719
Value of x is 6
Process id of shell is 20705
Child process:
Process id is 20720
Value of x is 6
Process id of parent 20719

kayon-pc11@kayon-pc11:~$
```

Wait.c


```c
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <sys/types.h>
#include <sys/wait.h>
int main(){
        int i, status;
        pid_t pid;
        pid = fork();
        if(pid < 0){
                printf("\nPembuatan proses gagal\n");
                exit(-1);
        }
        else if (pid > 0)
        {
                wait(NULL);
                printf("\nParent starts\nNomor genap:");
                for (i=2; i<=10;i+=2)
                        printf("%3d", i);
                printf("\nParent ends\n");
        }
        else if(pid == 0)
        {
                printf("Child start\nNomor ganjil:");
                for(i=1;i<10;i+=2)
                        printf("%3d", i);
                printf("\nChild ends\n");
        }
}
```

Exec.c


```c
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <sys/types.h>
int main(int argc, char*argv[]){
        pid_t pid;
        int i;

        if(argc != 3)
        {
                printf("\nInsufficient arguments to load program");
                printf("\nUsage: ./a.out <path> <cmd>\n"); exit (-1);
        }
        switch(pid = fork())
        {
        case -1:
                printf("fork failed");
                exit(-1);
        case 0:
                printf("child process\n");
                i= execl(argv[1], argv[2], 0);
                if(i < 0)
                {
                        printf("%s program not loaded using exec system call\n", argv[2]);
                        exit(-1);
                }
        default:
                wait(NULL);
                printf("child terminated\n");
                exit(0);
        }
}
```

Stat.c

```
#include <stdio.h>
#include <stdlib.h>
#include <sys/stat.h>
#include <time.h>
int main(int argc, char*argv[]){
        struct stat
        file; int n;
        if(argc != 2){
                printf("usage ./a.out <filename>\n"); exit(-1);}
        if(( n= stat( argv[1], &file)) == -1){
                perror(argv[1]);
                exit(-1);}
        printf("user id : %d\n", file.st_uid);
        printf("group id : %d\n", file.st_gid);
        printf("block size : %d\n", file.st_blksize);
        printf("block allocated : %d\n", file.st_blocks);
        printf("inode no.: %d\n", file.st_ino);
        printf("last accessed : %s", ctime(&(file.st_atime)));
        printf("last modified: %s", ctime(&(file.st_mtime)));
        printf("file size: %d bytes\n", file.st_size);
        printf("no. of links : %d\n", file.st_nlink);
        printf("permissions : ");
        printf( (S_ISDIR(file.st_mode))  ? "d" : "-");
        printf( (file.st_mode & S_IRUSR) ? "r" : "-");
        printf( (file.st_mode & S_IWUSR) ? "w" : "-");
        printf( (file.st_mode & S_IXUSR) ? "x" : "-");
        printf( (file.st_mode & S_IRGRP) ? "r" : "-");
        printf( (file.st_mode & S_IWGRP) ? "w" : "-");
        printf( (file.st_mode & S_IXGRP) ? "x" : "-");
        printf( (file.st_mode & S_IROTH) ? "r" : "-");
        printf( (file.st_mode & S_IWOTH) ? "w" : "-");
        printf( (file.st_mode & S_IXOTH) ? "x" : "-");
        printf("\n");
        if(file.st_mode & S_IFREG)
                printf("file type : Regular\n");
        if(file.st_mode &S_IFDIR)
                printf("file type : Directory\n");
}
```
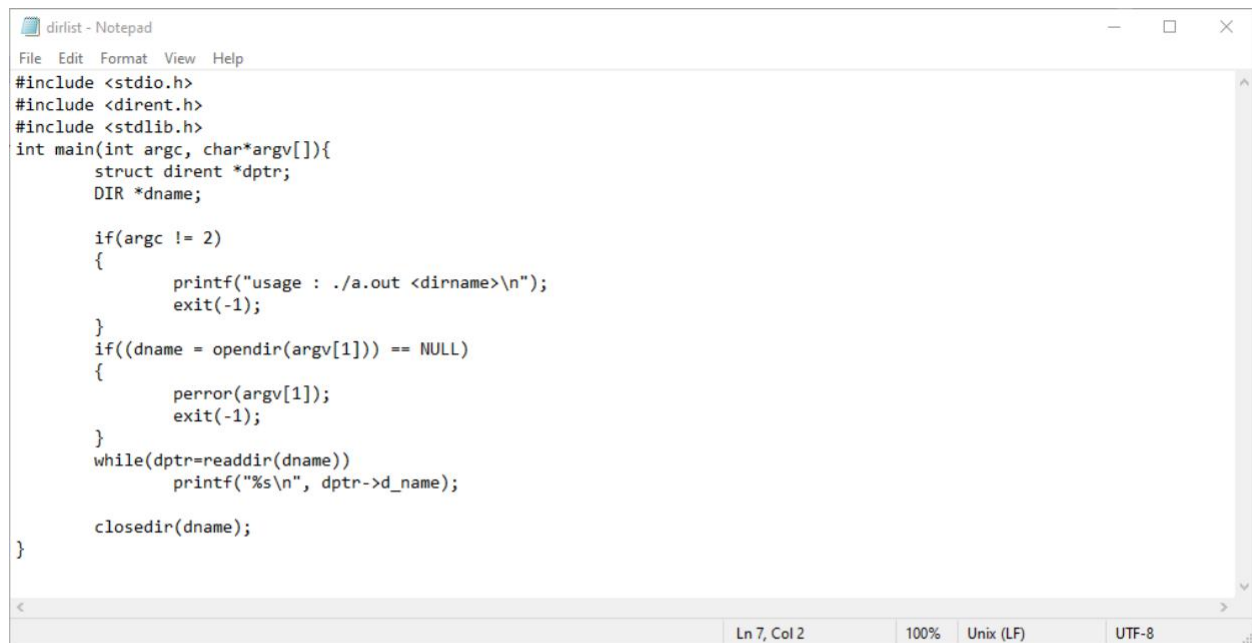
```
root@kayon-pc11:/home/kayon-pc11# ./a.out stat.c
user id : 1000
group id : 1000
block size : 4096
block allocated : 8
inode no.: 796237
last accessed : Fri Nov 22 17:24:12 2019
last modified: Tue Nov 19 15:36:42 2019
file size: 1363 bytes
no. of links : 1
permissions : -rw-r--r--
file type : Regular
```

Dirlist.c

```
dirlist - Notepad                                                    —    □    ×
File  Edit  Format  View  Help
#include <stdio.h>
#include <dirent.h>
#include <stdlib.h>
int main(int argc, char*argv[]){
        struct dirent *dptr;
        DIR *dname;

        if(argc != 2)
        {
                printf("usage : ./a.out <dirname>\n");
                exit(-1);
        }
        if((dname = opendir(argv[1])) == NULL)
        {
                perror(argv[1]);
                exit(-1);
        }
        while(dptr=readdir(dname))
                printf("%s\n", dptr->d_name);

        closedir(dname);
}

                                        Ln 7, Col 2    100%   Unix (LF)   UTF-8
```

```
root@kayon-pc11:/home/kayon-pc11# gcc dirlist.c
root@kayon-pc11:/home/kayon-pc11# ./a.out Documents
..
stat.c
wait.c
dirlist.c
fork.c
.
exec.c
root@kayon-pc11:/home/kayon-pc11#
```