1. To create ‘n’ children. When the children will terminate, display total cumulative time children spent in user and kernel mode.

```
#include <stdio.h>
#include <stdlib.h>
#include <sys/times.h>
#include <sys/types.h>

#define N 5

void child(int n) { int i, j; long
        t1, t2; struct tms cb1,
        cb2;

        t1 = times(&cb1); for(i=0;
        i<2;i++) { sleep(3 + i);
                for(j=0;j<1000000;j++);
        }
        t2 = times(&cb2);

        printf("child %d; real %u; user %u; sys %u;\n", n, t2-t1, cb2.tms_utime-
        cb1.tms_utime, cb2.tms_stime-cb1.tms_stime); exit(0);
}

int main() {
        int i; long t1, t2; struct
        tms cb1, cb2;

        t1 = times(&cb1); for(i = 0;
        i < N; i++) { if(fork() == 0) {
        child(i);
                }
        }
        for(i = 0; i < N; i++) {
                wait(0);
        }

        t2 = times(&cb2);
```

```
        printf("parent; real %u; user %u; sys %u;\n", t2-t1,
        cb2.tms_utimecb1.tms_utime, cb2.tms_stime-cb1.tms_stime); return 0;
}
```

Output :

```
akshay@akshu  :~/MSC CS/aos$ gcc program1.c
akshay@akshu  :~/MSC CS/aos$ ./a.out

child 4; real 701; user 1; sys 0; child 2;
real 701; user 1; sys 0; child 0; real 701;
user 1; sys 0; child 3; real 701; user 1; sys
0; child 1; real 701; user 0; sys 0; parent;
real 701; user 0; sys 0;
```

2.To generate parent process to write unnamed pipe and will read from it.

```
#include <stdio.h>
#include <stdlib.h>
#include <fcntl.h> #include
<string.h>

int main() { int fds[2];
        pipe(fds); if(fork() == 0)
        { char buff[1024];
                int n = read(fds[0], buff, 1024), i;
                printf("child read %d bytes: ", n);
                for(i=0; i<n; i++) printf("%c", buff[i]);
                printf("\n"); exit(0);
        }
        char *str = "hello, to child process";
        printf("parent writting %d bytes: %s\n", strlen(str), str);
        write(fds[1], str, strlen(str)); wait(0); return 0;
}
```

Output :

```
                :~/MSC CS/aos$ gcc program
                :~/MSC CS/aos$ ./a.out

akshay@akshu2.c
```

akshay@akshu

parent writting 23 bytes: hello, to child process child read
23 bytes: hello, to child process

3. To create a file with hole in it.

```
#include <fcntl.h>
#include <stdio.h> #include
<stdlib.h>

int main() { int fd = creat("./test", 0666);
        if(fd < 0) { perror("could create
        file"); exit(1); }
        printf("file created with fd: %d\n", fd); char* str =
        "hello\n";
        int i; for(i=0; i<5; i++) { write(fd, str,
        6); lseek(fd, 1024, SEEK_CUR);
        }
        close(fd); return
        0;
}
```

Output :

```
                    :~/MSC CS/aos$ gcc program
                    :~/MSC CS/aos$ ./a.out
```

akshay@akshu3.c
akshay@akshu

file created with fd: 3

4. Takes multiple files as Command Line Arguments and print their inode number.

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

```
int main(int argc, char** argv) { if(argc < 2) {
        printf("not enough args\n"); exit(0);
        }
        int i, fd; struct stat buff; for(i=1; i< argc; i++) {
        printf("opening %s\n", argv[i]); fd = open(argv[i],
        O_RDONLY); if (fd < 0) { printf("ERROR: could not open
        %s\n"); continue;
                }
                fstat(fd, &buff);
                printf("inode number %u\n", buff.st_ino); close(fd);
                printf("closing %s\n", argv[i]);
        }
}
```

Output :

```
                :~/MSC CS/aos$ gcc program
                :~/MSC CS/aos$ ./a.out
```

```
akshay@akshu4.c
akshay@akshu                                hello.txt run.txt

opening hello.txt inode
number 5780438 closing
hello.txt opening run.txt
inode number 5777124
closing run.txt
```

5. To handle the two-way communication between parent and child using pipe.

```
#include <stdio.h>
#include <fcntl.h>
#include <stdlib.h> #include
<string.h>

int main() { int to_child[2], to_parent[2]; pipe(to_child);
        pipe(to_parent); char buff[1024], buff2[1024]; char*
        kill = "CMD_KILL"; int count; if(fork() == 0) { while(1) {
        count = read(to_child[0], buff, 1024); if(strcmp(buff,
        kill) == 0) exit(0); strcpy(buff2, "hello, "); strcat(buff2,
        buff);
```

```
                write(to_parent[1], buff2, strlen(buff2) + 1); }
        } int len = 3;
        char *names[] = {"Akshay", "Rushi", "Ganesh"}; int i, j;
        for(i=0; i<len; i++) {
                write(to_child[1], names[i], strlen(names[i]) + 1); count
                = read(to_parent[0], buff, 1023); buff[1023] = '\0';
                        printf("recursive %d bytes from child: %s\n", count, buff);
        }
        write(to_child[1], kill, strlen(kill)+1); return 0;
}
```

Output :

```
                :~/MSC CS/aos$ gcc program
                :~/MSC CS/aos$ ./a.out
```

akshay@akshu5.c
akshay@akshu

recursive 14 bytes from child: hello, Akshay recursive 13
bytes from child: hello, Rushi recursive 14 bytes from
child: hello, Ganesh

6. Print the type of file where file name accepted through Command Line.

```
#include<stdio.h>
#include<stdlib.h>
#include<time.h>
#include<sys/types.h>
#include<sys/stat.h>


int main(int argc,char** argv){
        struct stat sb; if(argc < 2)
        { printf("no enough argument"); exit(0);
        }
        if(stat(argv[1],&sb)== -1)
```

```
        { perror("stat");
                exit(EXIT_FAIL
                URE);
        }
        printf("File tpye : "); switch(sb.st_mode &
        S_IFMT)
        { case S_IFBLK: printf("block device \n"); break;
                case S_IFCHR: printf("character device \n"); break;
                case S_IFDIR: printf("directory \n"); break;
                case S_IFIFO: printf("FIFO/pipe \n"); break;
                case S_IFLNK: printf("symlink \n"); break;
                case S_IFREG: printf("regular file \n"); break;
                case S_IFSOCK: printf("socket \n"); break;
                default:printf("unknown \n"); break;
        }
}
```

Output :

```
                :~/MSC CS/aos$ gcc program
                :~/MSC CS/aos$ ./a.out        blank
akshay@akshu6.c
akshay@akshu
```

File tpye : directory

7. To demonstrate the use of atexit() function.

```
#include <stdio.h> #include
<stdlib.h>

void function1() { printf("function 1
        Executes\n");
}
void function2() { printf("function 2
        Executes\n");
} int main() {
atexit(&function1);
```

```
atexit(&function2); return
0;
}
```

Output :

```
:~/MSC CS/aos$ gcc program
:~/MSC CS/aos$ ./a.out
```

akshay@akshu7.c
akshay@akshu

function 2 Executes function 1
Executes

8. Open a file goes to sleep for 15 seconds before terminating.

```
#include <stdio.h> #include
<fcntl.h>

int main(){ printf("File opening\n"); int fd =
        open("hello.txt",0666); printf("File goes to
        sleep 15 seconds\n"); sleep(15);
        printf("File terminated\n");
        close(fd); return 0;
}
```

Output :

```
:~/MSC CS/aos$ gcc program
:~/MSC CS/aos$ ./a.out
```

akshay@akshu8.c
akshay@akshu

File opening
File goes to sleep 15 seconds
File terminated
9. To print the size of the file.

```
#include<stdio.h>
#include<fcntl.h>

int main()
{ int fd; int size=0;
        char buf[512];
        fd=open("hello.txt",O_RDONLY); if(fd == NULL)
        printf("\n Unable to open this file");
        else printf("file open Successfully.\n");
        size=read(fd,buf,sizeof(buf));
        printf("Size of file is: %d bytes.\n",size);
        close(fd); return 0;
}
```

Output :

```
                :~/MSC CS/aos$ gcc program
                :~/MSC CS/aos$ ./a.out
```

akshay@akshu9.c
akshay@akshu

file open Successfully. Size of
file is: 32 bytes.

10. Read the current directory and display the name of the files, no of files in current directory.

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

int main(){ char
        cwd[1024];
        getcwd(cwd,
        sizeof(cwd));
        printf("cwd:
        %s\n", cwd); DIR
```

```
        *d =
        opendir(cwd);
        if(!d) exit(1);
        struct dirent *entry; int
        count = 0;
        while((entry = readdir(d)) != NULL) {
        printf("%s\n", entry->d_name); count++; }
        printf("%d entries found\n", count);
        closedir(d); return 0;
}
```

Output :

```
                :~/MSC CS/aos$ gcc program
                :~/MSC CS/aos$ ./a.out
```

akshay@akshu10.c
akshay@akshu

cwd: /home/akshay/MSC CS/aos
program4.c hello.txt
program7 program15.c
program12.c
program3
program11.c
program6 program1.c
program23.c
program5
program20.c
program13.c
program21.c
..
program22.c program5.c
program25.c program24.c
program14.c program6.c
run.txt program7.c
program2 a.out
program8.c
test

program9.c .vscode
program10
program1
program10.c
.
program8
program2.c
program3.c
blank program9
38 entries found

11. Write a C program to implement the following unix/linux command (use fork, pipe and exec system call) ls –l | wc –l

```
int main() { int fds[2];
        pipe(fds); if(fork()
        == 0) { close(1);
        dup(fds[1]);
        close(fds[0]);
        close(fds[1]);
        //child ls
                char *args[] = {"ls", "-l", 0};
                execvp(args[0], args); return 0;
        }
        if(fork() == 0) { close(0);
                dup(fds[0]);
                close(fds[0]);
                close(fds[1]); //child
                wc
                char *args[] = {"wc", "-l", 0};
                execvp(args[0], args); return;
        }
        close(fds[0]);
        close(fds[1]);
        wait(0);
        wait(0); return
        0;
}
```

Output :

:~/MSC CS/aos$ gcc program
:~/MSC CS/aos$ ./a.out

akshay@akshu11.c
akshay@akshu

37

12. Write a C program to display all the files from current directory which are created in particular month.

```
#include <stdio.h>
#include <time.h>
#include <unistd.h>
#include <sys/types.h>
#include <dirent.h>
#include <stdlib.h>
#include <fcntl.h> #include
<sys/stat.h>

int main() { int
        monthNum;
        printf("Enter month num (1-12) : ");
        scanf("%d", &monthNum);
        char cwd[1024];
        getcwd(cwd, sizeof(cwd));
        printf("cwd: %s\n", cwd); DIR
        *d = opendir(cwd); if(!d)
        exit(1); struct dirent *entry;
        int fd; struct stat buff;
        while((entry = readdir(d)) != NULL) { fd =
                open(entry->d_name, O_RDONLY);
                fstat(fd, &buff);
                if(monthNum == getMonth(buff.st_mtime)) { printf("%10s: ",
                        entry->d_name); printDate(buff.st_mtime);
                }
close(fd); } closedir(d);
return 0; }
```

```
int getMonth(time_t t) { char buff[100];
        struct tm* tmp = localtime(&t);
        strftime(buff, 100, "%m", tmp); int
        month;
        if(buff[0] == '0') month = buff[1] - '0'; else
        sscanf(buff, "%d", &month); return month;
}

int printDate(time_t t) { char buff[100]; struct tm*
        tmp = localtime(&t); strftime(buff, 100,
        "%d/%m/%y", tmp); printf(" %s\n", buff);
}
```

Output :

```
                    :~/MSC CS/aos$ gcc program
                    :~/MSC CS/aos$ ./a.out
```

akshay@akshu12.c
akshay@akshu

Enter month num (1-12) : 3 cwd:
/home/akshay/MSC CS/aos
program3.c: 30/03/22

13. Write a C program to display all the files from current directory whose size is greater that n Bytes, Where n is accept from user.

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

int main() { unsigned int size;
        printf("enter bytes : ");
        scanf("%u", &size); DIR *d
        = opendir("."); if(!d)
        exit(1);
```

```
        struct dirent *entry; struct
        stat buff;
        while((entry = readdir(d)) != NULL) { stat(entry->d_name, &buff);
                if(buff.st_size >= size) printf("%s\t %u\n", entry->d_name,
                buff.st_size);
        } closedir(d);
        return 0;
}
```

Output :

```
                :~/MSC CS/aos$ gcc program
                :~/MSC CS/aos$ ./a.out
```

akshay@akshu13.c
akshay@akshu
enter bytes : 512 program4.c
533 program11 18624
program7 19504
program12.c 1030 program3
19752 program6 20504
program1.c 769 program5
20384 program20.c 1155
program21.c 1312
.. 4096 program22.c
690 program5.c 790
program25.c 1772
program6.c 1882
program2 20088 a.out
16968 test 4126
program9.c 981 .vscode
4096 program10 20152
program1 20376
. 4096 program8
19472 program13
20840 blank 4096
program9 19672
program12 22160

14. Write a C program to implement the following unix/linux command i. ls –l > output.txt

```
int main() { int fd = creat("14-output.txt",
        0666); if(fork() == 0) { close(1); dup(fd);
                close(fd);
                char *args[] = {"ls", "-l", 0}; execvp(args[0], args);
        }
        close(fd);
        wait(0); return
        0;
}
```

Output :

```
                :~/MSC CS/aos$ gcc program
                :~/MSC CS/aos$ ./a.out
```

akshay@akshu14.c
akshay@akshu

15. Write a C program which display the information of a given file similar to given by the unix / linux command ls –l <file name>

```
#include <stdio.h> #include <stdlib.h> int main(int
argc, char *argv[]) { if (argc != 2) { printf("wrong
number of args\n"); exit(0);
        }
        if(fork() == 0) { char *args[] = {"ls", "-l", argv[1],
                0}; execvp(args[0], args);
        } wait(0);
        return 0;
}
```

Output :

```
                :~/MSC CS/aos$ gcc program       14
                :~/MSC CS/aos$ ./a.out hello.txt
```

akshay@akshu.c

akshay@akshu
-rw-rw-r-- 1 akshay akshay 32 May 16 15:26 hello.txt
20. Write a C program which receives file names as command line arguments and display those filenames in ascending order according to their sizes.
i)e.g $ a.out a.txt b.txt c.txt, …)

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

struct entry { char
        *name;
        unsigned long size;
};

void bubbleSort(struct entry *entries, int n) {
        int i, j; struct entry temp; for(i=0; i<n; i++) { for(j=0;
        j<(n-i-1); j++) { if(entries[j].size > entries[j+1].size) {
        temp.name = entries[j].name; temp.size =
        entries[j].size;

                                entries[j].name = entries[j+1].name; entries[j].size =
                                entries[j+1].size;

                                entries[j+1].name = temp.name; entries[j+1].size =
                                temp.size;
                        }
                }
        }
}

int main(int argc, char** argv) { if(argc < 2) {
        printf("not enough args\n");
                exit(0);
        }
```

```
        struct entry* entries = malloc(sizeof(struct entry) * (argc-1)); int fd,
        i; struct stat buff; for(i=1; i<argc; i++) { fd = open(argv[i],
        O_RDONLY); if (fd < 0) { printf("ERROR: could not open %s\n");
        exit(1);
                }
                fstat(fd, &buff); entries[i-1].name =
                argv[i]; entries[i-1].size =
                buff.st_size; close(fd);
        }
        bubbleSort(entries, argc-1); for(i=0; i<(argc-1); i++) { printf("name: %s\tsize:
        %u\n", entries[i].name, entries[i].size);
        } return
        0;
}
```

Output :

```
akshay@akshu  :~/MSC CS/aos$ gcc program       20.c
akshay@akshu  :~/MSC CS/aos$ ./a.out hello.txt run.txt test

name: hello.txt size: 32 name:
run.txt   size: 39 name: test
size: 4126
```

21. Write a C program which create a child process which catch a signal sighup, sigint and sigquit. The Parent process send a sighup or sigint signal after every 3 seconds, at the end of 30 second parent send sigquit signal to child and child terminates my displaying message "My DADDY has Killed me!!!".

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

void catch_int() { printf("CHILD:
        SIGINT\n");
}
```

```
void catch_hup() { printf("CHILD:
        SIGHUP\n");
}

void catch_quit() { printf("CHILD:
SIGQUIT\n"); exit(0); }

int main() { int child; if((child = fork()) == 0) {
        signal(SIGINT, catch_int);
        signal(SIGHUP, catch_hup);
        signal(SIGQUIT, catch_quit); while(1);
        }
        int i; for(i=0; i<5; i++)
        { sleep(1);
                        kill(child, i&1 ? SIGINT : SIGHUP);
        } sleep(1); kill(child,
        SIGQUIT); wait(0);
        printf("PARENT: child exit\n"); return 0;
}
```

Output :

```
                :~/MSC CS/aos$ gcc program
                :~/MSC CS/aos$ ./a.out
```

akshay@akshu21.c
akshay@akshu

```
CHILD: SIGHUP
CHILD: SIGINT
CHILD: SIGHUP
CHILD: SIGINT
CHILD: SIGHUP
CHILD: SIGQUIT
PARENT: child exit
```

23. Write a C Program that demonstrates redirection of standard output to a file.

```
#include <stdio.h> #include
<fcntl.h>
```

```
int main() { int fd = creat("stdout-23.c",
        0666); close(1); dup(fd); close(fd);
        printf("hello world"); return 0;
}
```

Output :

```
                :~/MSC CS/aos$ gcc program
                :~/MSC CS/aos$ ./a.out
```

akshay@akshu23.c

akshay@akshu

24. Write a program that illustrates how to execute two commands concurrently with a pipe.

```
int main() { int fds[2];
        pipe(fds); if(fork()
        == 0) { close(1);
        dup(fds[1]);
        close(fds[0]);
        close(fds[1]);
        //child ls
                char *args[] = {"ls", "-l", 0};
                execvp(args[0], args); return 0;
        }
        if(fork() == 0) { close(0);
                dup(fds[0]);
                close(fds[0]);
                close(fds[1]); //child
                wc
                char *args[] = {"wc", "-l", 0};
                execvp(args[0], args); return;
        }
        close(fds[0]);
        close(fds[1]);
        wait(0);
        wait(0); return
        0;
}
```

Output :

```
:~/MSC CS/aos$ gcc program
:~/MSC CS/aos$ ./a.out
```

akshay@akshu24.c
akshay@akshu

47

25. Write a C program that illustrates suspending and resuming processes using signals.

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

int child;

void catch_int(int s) { printf("\n\nPARENT: Suspending
        child\n"); int r = kill(child, SIGSTOP); if(r == -1)
        perror("kill SIGSTOP: ");
        printf("Press any key -1 to exit, anything else to resume: "); int opt;
        scanf("%d", &opt); if(opt == -1) { printf("PARENT: killing child\n"); r =
        kill(child, SIGKILL);
                if(r == -1) perror("kill SIGKILL: ");
        }
        printf("PARENT: Resuming child\n"); r =
        kill(child, SIGCONT);
        if(r == -1) perror("kill SIGCONT: ");
}

void catch_cont(int s) { printf("CHILD:
        SIGCONT");
}

void catch_int_child(int s) { printf("CHILD:
        SIGINT");
}
```

```
void main() { child =
        fork(); if(child ==
        0) {
        signal(SIGCONT,
        catch_cont);
        signal(SIGINT,
        catch_int_child);
        int num = 0;
        while(1) {
        printf("I AM
        CHILD: %d\n",
        num++);
        sleep(1);
                }
        }
        signal(SIGINT, catch_int); int r; while(1) { r =
        wait(0); if(r == -1 || r == child) { printf("PARENT:
        exit, %d\n", r);
                        exit(0);
                } sleep(1);
        }
}
```

Output :

```
                :~/MSC CS/aos$ gcc program
                :~/MSC CS/aos$ ./a.out
```

akshay@akshu25.c
akshay@akshu

```
I AM CHILD: 0
I AM CHILD: 1
I AM CHILD: 2
I AM CHILD: 3
I AM CHILD: 4
```

Q.Write a C program to read all txt files (that is files that ends with .txt) in the current directory and merge them all to one txt file and returns a file descriptor for the new file

```
#include<stdio.h>
#include<fcntl.h>
#include<dirent.h>
#include<string.h>
int openf(char *);
int main()
{
DIR *i;
struct dirent *k;
int j,l,s,flag=1,fd,fd1;
char *m = "txt.";
char ch;
fd1 = openf("new.txt");
i = opendir("./test");
printf("\n%d\n",i);

while(( k = readdir(i)) !=  NULL)
{
flag=1;
l=strlen(k->d_name);
for(j = l-1,s=0;j >= l-4,s < 4 ;j--,s++ )
{
if((k->d_name[j])!= m[s])
{
flag=0;
```

```
goto q;
}

}
q:
if(flag==1)
{
fd = openf(k->d_name);
while(read(fd, &ch, 1))
write(fd1,&ch,1);
close(fd);
}
}
printf("\n");
close(fd1);
}
int openf(char * s)
{
int fd;
fd = open(s,O_RDWR|O_CREAT,0666);
if(fd < 1)
perror("open error");
else
printf("input file open succesfull fd =
%d\n",fd);
return fd;
}
```

Q.Write a C program to Identify the type (Directory, character device, Block device, Regular file, FIFO or pipe, symbolic link or socket) of given file using stat() system call.

```
#include <stdio.h>
#include <sys/stat.h>
#include <stdlib.h>
int main(int argc, char **argv) {
        struct stat *st;
        st = (struct stat *)
malloc(sizeof(struct stat));

        stat(argv[1], st);

        if(S_ISDIR(st->st_mode))
                printf("file type is :
Directory\n");
        else if (S_ISCHR(st->st_mode))
                printf("file type is :
Character device\n");
        else if (S_ISBLK(st->st_mode))
                printf("file type is :
Block device\n");
        else if (S_ISREG(st->st_mode))
                printf("file type is :
Regular file\n");
        else if (S_ISFIFO(st->st_mode))
                printf("file type is :
FIFO or Pipe\n");
        else if (S_ISLNK(st->st_mode))
```

```
                printf("file type is :
Symbolic link\n");
        else if (S_ISSOCK(st->st_mode))
                printf("file type is :
Socket\n");
        return 0;
}
```