

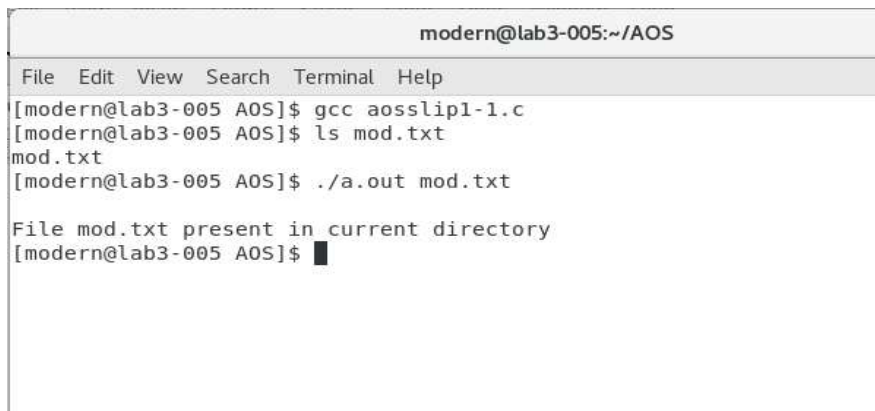
/*Write a C program to find whether a given file is present in current directory or not.*/

Solution:→

```
#include<stdio.h>
#include<dirent.h>
#include<stdlib.h>
#include<string.h>
#include<unistd.h>
```

```
int main(int argc,char *argv[])
{
    DIR *dirptr;
    struct dirent *entry;
    int found = 0;
    char curDir[20];
    getcwd(curDir,20);
    if(argc<2)
    {
        printf("\n Insufficient arguments\n");
        exit(1);
    }
    dirptr = opendir(curDir);
    while((entry = readdir(dirptr))!=NULL)
    {
        if(strcmp(entry->d_name,argv[1])==0)
        {
            printf("\nFile %s present in current directory\n",argv[1]);
            found=1;
            break;
        }
    }
    closedir(dirptr);
    if(found==0)
        printf("\nnFile %s not present in current directory\n",argv[1]);
    return 0;
}
```

output:



```
modern@lab3-005:~/AOS
File Edit View Search Terminal Help
[modern@lab3-005 AOS]$ gcc aosslipl-1.c
[modern@lab3-005 AOS]$ ls mod.txt
mod.txt
[modern@lab3-005 AOS]$ ./a.out mod.txt
File mod.txt present in current directory
[modern@lab3-005 AOS]$
```

/*Write a C program that a string as an argument and return all the files that begins with that name in the current directory. For example > ./a.out d will return all file names that begins with d.*/

Solution:→

```
#include<stdio.h>
#include<dirent.h>
#include<stdlib.h>
#include<string.h>
#include<unistd.h>
```

```
int startsWith(char *fileName, char *start)
{
    int i,len = strlen(start);
    for(i=0;i<len;i++)
    {
        if(fileName[i]==start[i])
            continue;
        break;
    }
    if(i==len)
        return 0;
    return 1;
}
```

```
int main(int argc,char *argv[])
{
    DIR *dirptr;
    struct dirent *entry;
    char curDir[20];

    getcwd(curDir,20);
    if(argc<2)
    {
        printf("\n Insufficient arguments\n");
        exit(1);
    }
    dirptr = opendir(curDir);
    while((entry = readdir(dirptr))!=NULL)
    {
        if(startsWith(entry->d_name,argv[1])==0)
            printf("\n%s",entry->d_name);
    }
    closedir(dirptr);
    return 0;
}
```

```
modern@lab3-005:~/AOS
File Edit View Search Terminal Help
[modern@lab3-005 AOS]$ vi aossli2-1.c
[modern@lab3-005 AOS]$ gcc aossli2-1.c
[modern@lab3-005 AOS]$ ./a.out d

dd.txt
dip.txt[modern@lab3-005 AOS]$ █
```

/*Write a C program to find file properties such as inode number, number of hard link, File permissions, File size, File access and modification time and so on of a given file using stat() system call.*/

Solution:→

```
#include<sys/stat.h>
#include<string.h>
#include<stdio.h>
#include<stdlib.h>
#include<time.h>
#include<sys/types.h>
#include<pwd.h>
#include<grp.h>
int main(int argc,char *argv[])
{
    struct stat s;
    struct tm *timeinfo;
    struct passwd *pw;
    struct group*gr;
    char filetype,perm,*date;
    int i;
    memset(&s,0,sizeof(s));

    if(argc<2)
    {
        printf("Insufficient arguments\n");
        exit(1);
    }
    printf("\nFile size \t inode\n");
    for( i=1;i<argc;i++)
    {
        printf("\n");
        stat(argv[i],&s);
        if((s.st_mode & S_IFMT)==S_IFREG) filetype = 'R';
        else if((s.st_mode & S_IFMT)==S_IFSOCK) filetype = 'S';
```

```

else if((s.st_mode & S_IFMT)==S_IFLNK) filetype = 'L';
else if((s.st_mode & S_IFMT)==S_IFBLK) filetype = 'B';
else if((s.st_mode & S_IFMT)==S_IFDIR) filetype = 'D';
else if((s.st_mode & S_IFMT)==S_IFCHR) filetype = 'C';
else if((s.st_mode & S_IFMT)==S_IFIFO) filetype = 'F';
printf("%s\t%d\tC\t%d\t%d",argv[i],s.st_ino,filetype,s.st_size,s.st_nlink);

date = ctime(&s.st_atime);
timeinfo = localtime(&s.st_atime);
printf("\nmonth=%d\n",timeinfo->tm_mon);
printf("\nFile access time = %s",date);
printf("\nFile access time = %s",ctime(&s.st_mtime));
printf("\nFile access time = %s",ctime(&s.st_ctime));
pw = getpwuid(s.st_uid);
gr = getgrgid(s.st_gid);
printf("\n user = %s",pw->pw_name);
printf("\n group = %s",gr->gr_name);
printf((s.st_mode & S_IRUSR)?"r":"-");
printf((s.st_mode & S_IWUSR)?"w":"-");
printf((s.st_mode & S_IXUSR)?"x":"-");
printf((s.st_mode & S_IRGRP)?"r":"-");
printf((s.st_mode & S_IWGRP)?"w":"-");
printf((s.st_mode & S_IXGRP)?"x":"-");
printf((s.st_mode & S_IROTH)?"r":"-");
printf((s.st_mode & S_IWOTH)?"w":"-");
printf((s.st_mode & S_IXOTH)?"x":"-");
}
return 0;
}

```

output:

```

modern@lab3-005:~/AOS
File Edit View Search Terminal Help
[modern@lab3-005 AOS]$ vi aossli3-1.c
[modern@lab3-005 AOS]$ ./a.out dip.txt

File size      inode
dip.txt 100761104      R      23      1
month=3

File access time = Wed Apr 26 11:32:36 2023
File access time = Wed Apr 26 11:32:36 2023
File access time = Wed Apr 26 11:32:36 2023

user = modern
group = modernrw-rw-r-- [modern@lab3-005 AOS]$ █

```

/*Write a C program to find file properties such as inode

number, number of hard link, File permissions, File size, File access and modification time and so on of a given file using fstat() system call.*/

Solution:→

```
#include<sys/stat.h>
#include<string.h>
#include<stdio.h>
#include<stdlib.h>
#include<time.h>
#include<sys/types.h>
#include<pwd.h>
#include<grp.h>
#include<fcntl.h>
```

```
int main(int argc,char *argv[])
{
    struct stat s;
    struct tm *timeinfo;
    struct passwd *pw;
    struct group*gr;
    char filetype,perm,*date;
    memset(&s,0,sizeof(s));
    if(argc<2)
    {
        printf("Insufficient arguments\n");
        exit(1);
    }
    for(int i=1;i<argc;i++)
    {
        printf("\n");
        int fd = open(argv[i],O_RDONLY);
        fstat(fd,&s);
        if((s.st_mode & S_IFMT)==S_IFREG) filetype = 'R';
        else if((s.st_mode & S_IFMT)==S_IFSOCK) filetype = 'S';
        else if((s.st_mode & S_IFMT)==S_IFLNK) filetype = 'L';
        else if((s.st_mode & S_IFMT)==S_IFBLK) filetype = 'B';
        else if((s.st_mode & S_IFMT)==S_IFDIR) filetype = 'D';
        else if((s.st_mode & S_IFMT)==S_IFCHR) filetype = 'C';
        else if((s.st_mode & S_IFMT)==S_IFIFO) filetype = 'F';
        printf("\nfile%s\ninode=%ld\nfiletype=%c\nfilesize=%ld\nnumber of
links=%ld",argv[i],s.st_ino,filetype,s.st_size,s.st_nlink);
        date = ctime(&s.st_atime);
        timeinfo = localtime(&s.st_atime);
        printf("\nmonth=%d\n",timeinfo->tm_mon);
        printf("\nFile access time = %s",date);
        printf("\nFile access time = %s",ctime(&s.st_mtime));
        printf("\nFile access time = %s",ctime(&s.st_ctime));
```

```

        pw = getpwuid(s.st_uid);
        gr = getgrgid(s.st_gid);
        printf("\n user = %s",pw->pw_name);
        printf("\n group = %s",gr->gr_name);
        printf((s.st_mode & S_IRUSR)?"r":"-");
        printf((s.st_mode & S_IWUSR)?"w":"-");
        printf((s.st_mode & S_IXUSR)?"x":"-");
        printf((s.st_mode & S_IRGRP)?"r":"-");
        printf((s.st_mode & S_IWGRP)?"w":"-");
        printf((s.st_mode & S_IXGRP)?"x":"-");
        printf((s.st_mode & S_IROTH)?"r":"-");
        printf((s.st_mode & S_IWOTH)?"w":"-");
        printf((s.st_mode & S_IXOTH)?"x":"-");
    }
    return 0;
}

```

output:

```

modern@lab3-005:~/AOS
File Edit View Search Terminal Help
[modern@lab3-005 AOS]$ vi aossli4-1.c
[modern@lab3-005 AOS]$ ./a.out dip.txt

File size      inode
dip.txt 100761104      R      23      1
month=3

File access time = Wed Apr 26 11:32:36 2023
File access time = Wed Apr 26 11:32:36 2023
File access time = Wed Apr 26 11:32:36 2023

user = modern
group = modernrw-rw-r--[modern@lab3-005 AOS]$ █

```

/* Write a C program to create an unnamed pipe.

The child process will write following three messages to pipe and parent process display it.

Message1 =

Message2 =

Message3 =

*/

Solution:→

```

#include<stdio.h>
#include<unistd.h>
#include<stdlib.h>
#define MSGSIZE 16

```

```

char *msg1 = "Hello World";
char *msg2 = "Hello SPPU";
char *msg3 = "Linux is funny";

```

```

int main()
{
    int fd[2],i;
    char buff[MSGSIZE];

    if(pipe(fd)<0)
        exit(1);

    write(fd[1],msg1,MSGSIZE);
    write(fd[1],msg2,MSGSIZE);
    write(fd[1],msg3,MSGSIZE);

    for(i=0;i<3;i++)
    {
        read(fd[0],buff,MSGSIZE);
        printf("\n%s",buff);
    }
    return 0;
}

```

output

```

modern@lab3-005:~/AOS
File Edit View Search Terminal Help
[modern@lab3-005 AOS]$ vi aossli5-1.c
[modern@lab3-005 AOS]$ ./a.out
Hello World
Hello SPPU
Linux is funny[modern@lab3-005 AOS]$

```

/*
Write a C program to map a given file in memory and display the contain of mapped file in reverse.
*/

*/

Solution:→

```

#include<stdio.h>
#include<stdlib.h>
#include<sys/mman.h>
#include<unistd.h>

```

```

#include<sys/stat.h>
#include<sys/types.h>
#include<fcntl.h>
#include<string.h>
#include<malloc.h>

char *addr;

int main()
{
    int fd,len,stats;
    char *rev;
    struct stat st;
    memset(&st,0,sizeof(st));

    fd = open("dip.txt",O_RDONLY);
    stats = fstat(fd,&st);
    len = st.st_size;

    if((addr = mmap(NULL, len, PROT_READ, MAP_PRIVATE,fd,0))==MAP_FAILED)
        printf("\nError in mmap");
    printf("\nmmap=\n%s",addr); // display mapped file

    printf("\nmapped file in reverse order\n");
    rev = addr + strlen(addr);
    while(rev != addr)
    {
        printf("%c",*rev);
        rev--;
    }
    printf("%c",*rev);
    return 0;
}

```

output:


```
modern@lab3-005:~/AOS
File Edit View Search Terminal Help
[modern@lab3-005 AOS]$ cat dip.txt
This is Modern College
[modern@lab3-005 AOS]$ vi aosslip6-1.c
[modern@lab3-005 AOS]$ gcc aosslip6-1.c
[modern@lab3-005 AOS]$ ./a.out

map=
This is Modern College

mapped file in reverse order

egelloC nredoM si sihT[modern@lab3-005 AOS]$
```

/* Write a C program to create a file with hole in it. */

Solution:→

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

int main()
{
    int fd;
    char *msg1="Welcome to";
    char *msg2 = "WELCOME TO";
    fd = open("p1.txt",O_CREAT | O_WRONLY);
    if(fd<0)
    {
        printf("\nfailed to create file\n");
        exit(1);
    }
    chmod("p1.txt",0777);
    write(fd,msg1,10);
    lseek(fd,1024L,SEEK_CUR);
    write(fd,msg2,10);
    close(fd);
    return 0;
}
```

output:

```
modern@lab3-005:~/AOS
File Edit View Search Terminal Help
[modern@lab3-005 AOS]$ vi aossli7-1.c
[modern@lab3-005 AOS]$ gcc aossli7-1.c
[modern@lab3-005 AOS]$ ./a.out
[modern@lab3-005 AOS]$ vi p1.txt
```

the p1.txt file with hole

```
modern@lab3-005:~/AOS
File Edit View Search Terminal Help
Welcome to
"p1.txt" 11. 1045C
```

/* Write a C program to get and set the resource limits such as files, memory associated with a process.*/

Solution:→

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

int main()
{
    struct rlimit limit;
    if(getrlimit(RLIMIT_NPROC,&limit)<0)
        printf("getrlimit error\n");
    printf("No. of extant process = [%10ld][%10ld]\n",limit.rlim_max,limit.rlim_cur);

    if(getrlimit(RLIMIT_CPU,&limit)<0)
        printf("getrlimit error\n");
```

```

        printf("limit on amount of CPU time that process can consume =
[%ld][%ld]\n",limit.rlim_max,limit.rlim_cur);

        if(getrlimit(RLIMIT_DATA,&limit)<0)
            printf("getrlimit error\n");
        printf("max.size of process's data segment =
[%ld][%ld]\n",limit.rlim_max,limit.rlim_cur);

        if(getrlimit(RLIMIT_FSIZE,&limit)<0)
            printf("getrlimit error\n");
        printf("max. size in bytes of files that process may create =
[%ld][%ld]\n",limit.rlim_max,limit.rlim_cur);

        if(getrlimit(RLIMIT_LOCKS,&limit)<0)
            printf("getrlimit error\n");
        printf("limit on locks = [%ld][%ld]\n",limit.rlim_max,limit.rlim_cur);

        if(getrlimit(RLIMIT_MEMLOCK,&limit)<0)
            printf("getrlimit error\n");
        printf("max. no. of bytes of memory that can be locked in RAM =
[%ld][%ld]\n",limit.rlim_max,limit.rlim_cur);

        if(getrlimit(RLIMIT_MSGQUEUE,&limit)<0)
            printf("getrlimit error\n");
        printf("msg queue = [%ld][%ld]\n",limit.rlim_max,limit.rlim_cur);

    return 0;
}

```

output

```

modern@lab3-005:~/AOS
File Edit View Search Terminal Help
[modern@lab3-005 AOS]$ vi aossli8-1.c
[modern@lab3-005 AOS]$ gcc aossli8-1.c
[modern@lab3-005 AOS]$ ./a.out
No. of extant process = [ 63740][ 4096]
limit on amount of CPU time that process can consume = [-1][-1]
max.size of process's data segment = [-1][-1]
max. size in bytes of files that process may create = [-1][-1]
limit on locks = [-1][-1]
max. no. of bytes of memory that can be locked in RAM = [65536][65536]
msg queue = [819200][819200]
[modern@lab3-005 AOS]$ █

```

/*Write a C program to display as well as resets the environment variable such as path, home, root etc.*/

Solution:→

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

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

```
int main()
```

```
{
```

```
    char *path = getenv("PATH");
```

```
    if(path)
```

```
        printf("\npath=%s\n",path);
```

```
    else printf("\nvar not found\n");
```

```
    char *home = getenv("HOME");
```

```
    if(home)
```

```
        printf("\nhome=%s\n",home);
```

```
    else printf("\nvar not found\n");
```

```
    char *shell = getenv("SHELL");
```

```
    if(shell)
```

```
        printf("\nshell=%s\n",shell);
```

```
    else printf("\nvar not found\n");
```

```
    setenv("HOME","/home/AOS",1);
```

```
    home = getenv("HOME");
```

```
    if(home)
```

```
        printf("\nhome=%s\n",home);
```

```
    else printf("\nvar not found\n");
```

```
    return 0;
```

```
}
```

output:

```
modern@lab3-005:~/AOS
File Edit View Search Terminal Help
[modern@lab3-005 AOS]$ vi aossli9-1.c
[modern@lab3-005 AOS]$ gcc aossli9-1.c
[modern@lab3-005 AOS]$ ./a.out

path=/usr/lib64/qt-3.3/bin:/home/modern/perl5/bin:/usr/local/bin:/usr/local/sbin:/usr/bin:/usr/sbin:/bin:/sbin:/home/modern/.local/bin:/home/modern/bin

home=/home/modern

shell=/bin/bash

home=/home/AOS
[modern@lab3-005 AOS]$
```

/*Write a C program to create variable length arrays using alloca() system call.*/

Solution:→

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

int main()
{
    //    int *arr1 = malloc(20);
    int *arr1 = alloca(20);
    memset(arr1,0,20);
    printf("\narr1 size = %lu\n",sizeof(arr1));
    //    free(arr1);
    return 0;
}
```

output:

```
[modern@lab3-005 AOS]$ vi aossilip11-1.c
[modern@lab3-005 AOS]$ gcc aossilip11-1.c
[modern@lab3-005 AOS]$ ./a.out

arr1 size = 8
[modern@lab3-005 AOS]$ █
```

/*Write a C program to send SIGALRM signal by child process to parent process and parent process make a provision to catch the signal and display alarm is fired.(Use Kill, fork, signal and sleep system call)*/

Solution:→

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

static void my_alarm(int signo)
{
    printf("\n in signal handler");
    alarm(1);
}

int main()
{
```

```

int i;
pid_t pid;
signal(SIGALRM,my_alarm);

if((pid=fork())<0)
    printf("\nfork error");
if(pid==0)
{
    printf("\n child");
    alarm(2);
    kill(getppid(),SIGALRM);
}
// alarm(2);
else
{
    printf("\nparent");
    for(i=1;;i++)
    {
        printf("\n inside main");
        sleep(1);
    }
}
return 0;
}

```

output:

.

```

[modern@lab3-005 A0S]$ vi aossilip12-1.c
[modern@lab3-005 A0S]$ gcc aossilip12-1.c
[modern@lab3-005 A0S]$ ./a.out

```

parent

```

inside main
in signal handler
child inside main
in signal handler
inside main
in signal handler
inside main
in signal handler
inside main
in signal handler
inside main
in signal handler
inside main
in signal handler

```

^C

```

[modern@lab3-005 A0S]$ █

```

/* Write a C program to create an unnamed pipe. Write following three messages to pipe and display it.

Message1 =
Message2 =
Message3 =
*/

Solution:→

```
#include<stdio.h>
#include<unistd.h>
#include<stdlib.h>
#define MSGSIZE 16

char *msg1 = "Hello World";
char *msg2 = "Hello SPPU";
char *msg3 = "Linux is funny";

int main()
{
    int fd[2],i;
    char buff[MSGSIZE];

    if(pipe(fd)<0)
        exit(1);

    write(fd[1],msg1,MSGSIZE);
    write(fd[1],msg2,MSGSIZE);
    write(fd[1],msg3,MSGSIZE);

    for(i=0;i<3;i++)
    {
        read(fd[0],buff,MSGSIZE);
        printf("\n%s",buff);
    }
    return 0;
}
```

output:

```
modern@lab3-005:~/AOS
File Edit View Search Terminal Help
[modern@lab3-005 AOS]$ vi aossli14-1.c
[modern@lab3-005 AOS]$ ./a.out

Hello World
Hello SPPU
Linux is funny[modern@lab3-005 AOS]$
```

/* 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. */
Solution:→

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

int main(int argc,char *argv[])
{
    struct stat s;
    char filetype;
    memset(&s,0,sizeof(s));

    if(argc<2)
    {
        printf("Insufficient arguments\n");
        exit(1);
    }
    stat(argv[1],&s);
    if((s.st_mode & S_IFMT)==S_IFREG) filetype = 'R';
    else if((s.st_mode & S_IFMT)==S_IFSOCK) filetype = 'S';
    else if((s.st_mode & S_IFMT)==S_IFLNK) filetype = 'L';
    else if((s.st_mode & S_IFMT)==S_IFBLK) filetype = 'B';
    else if((s.st_mode & S_IFMT)==S_IFDIR) filetype = 'D';
    else if((s.st_mode & S_IFMT)==S_IFCHR) filetype = 'C';
    else if((s.st_mode & S_IFMT)==S_IFIFO) filetype = 'F';
    printf("%s\t%c\n",argv[1],filetype);
    return 0;
}
```


output

```
modern@lab3-005:~/AOS
File Edit View Search Terminal Help
[modern@lab3-005 AOS]$ gcc aosslip15-1.c
[modern@lab3-005 AOS]$ ./a.out dip.txt
dip.txt R
[modern@lab3-005 AOS]$ █
```

/* Write a C program that catches the ctrl-c (SIGINT) signal for the first time and display the appropriate message and exits on pressing ctrl-c again. */

Solution:→

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

void handle_sigint(int sig)
{
    printf("\ncaught signal %d\n",sig);
    signal(SIGINT, SIG_DFL);
}

int main()
{
    signal(SIGINT,handle_sigint);
    while(1)
    {
        printf("hello world\n");
        sleep(1);
    }
    return 0;
}
```

output:

```

[modern@lab3-005 AOS]$ vi aosslipl6-1.c
[modern@lab3-005 AOS]$ gcc aosslipl6-1.c
[modern@lab3-005 AOS]$ ./a.out
hello world
hello world
hello world
^C
caught signal 2
hello world
hello world
hello world
^Xhello world
hello world
hello world
hello world
^Xhello world
^C
[modern@lab3-005 AOS]$ █

```

/* Write a C program to display the given message times. (make a use of setjmp and longjmp system call) */

Solution:→

```

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

static jmp_buf jmpbuffer;

static void f1()
{
    printf("\nInside f1(): before longjmp");
    longjmp(jmpbuffer,1);
    printf("\nInside f1(): after longjmp");
}

int main()
{
    if(setjmp(jmpbuffer)!=0)
        printf("\nInside main():after longjmp\n");
    else
    {
        printf("\nInside main: calling f1()");
        f1();
    }

    return 0;
}

```

output:

```
modern@lab3-005:~/AOS
File Edit View Search Terminal Help
[modern@lab3-005 AOS]$ vi aosslipl7-1.c
[modern@lab3-005 AOS]$ gcc aosslipl7-1.c
[modern@lab3-005 AOS]$ ./a.out

Inside main: calling f1()
Inside f1(): before longjmp
Inside main():after longjmp
[modern@lab3-005 AOS]$ █
```

/* Write a C program to display the last access and modified time of a given file. */

Solution:→

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

int main(int argc,char *argv[])
{
    struct stat s;
    struct tm *timeinfo;
    memset(&s,0,sizeof(s));
    char *date;

    if(argc<2)
    {
        printf("Insufficient arguments\n");
        exit(1);
    }
    stat(argv[1],&s);
    date = ctime(&s.st_atime);
    timeinfo = localtime(&s.st_atime);
    printf("\nFile modification time = %s",ctime(&s.st_mtime));
    printf("\nFile access time = %s",ctime(&s.st_atime));
```

```
    return 0;
}
```

output:

```
modern@lab3-005:~/AOS
File Edit View Search Terminal Help
[modern@lab3-005 AOS]$ vi aosslipl8-1.c
[modern@lab3-005 AOS]$ gcc aosslipl8-1.c
[modern@lab3-005 AOS]$ ./a.out dip.txt

File modification time = Wed Apr 26 14:11:44 2023

File access time = Wed Apr 26 14:03:39 2023
[modern@lab3-005 AOS]$ █
```

/* Write a C program to move the content of file1.txt to file2.txt and remove the file1.txt from directory. */
Solution:→

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

int main()
{
    char ch;
    int fd1 = open("dip.txt",O_RDONLY);
    int fd2 = creat("dip1.txt",O_CREAT | O_WRONLY);
    while((read(fd1,&ch,1)!=0))
        write(fd2,&ch,1);
    close(fd1);
    close(fd2);
    unlink("dip.txt");
    return 0;
}
```

output:

```
modern@lab3-005:~/AOS
File Edit View Search Terminal Help
[modern@lab3-005 AOS]$ gcc aossli19-1.c
[modern@lab3-005 AOS]$ vi aossli19-1.c
[modern@lab3-005 AOS]$ gcc aossli19-1.c
[modern@lab3-005 AOS]$ ./a.out
[modern@lab3-005 AOS]$ cat dip.txt
cat: dip.txt: No such file or directory
[modern@lab3-005 AOS]$ cat dipl.txt
cat: dipl.txt: Permission denied
[modern@lab3-005 AOS]$ vi aossli19-1.c
[modern@lab3-005 AOS]$ █
```

/* Write a C program that print the exit status of a terminated child process.*/

Solution:→

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

void pr_exit(int status)
{
    if(WIFEXITED(status))
        printf("\nnormal termination\nexit status = %d\n",WEXITSTATUS(status));
    else if(WIFSIGNALED(status))
        printf("\nabnormal termination\nsignal number =
%d%s\n",WTERMSIG(status),

#ifdef WCOREDUMP
        WCOREDUMP(status)? "(Core file generated)": "");
#else
        "");
#endif
    else if(WIFSTOPPED(status))
        printf("\nchild stopped \nsignal number = %d\n",WSTOPSIG(status));
}

int main()
{
    pid_t pid;
```

```

int status;

if((pid=fork())<0)
    printf("fork error");
else if(pid==0)    //child
    exit(7);

if(wait(&status) != pid)    //wait for child
    printf("wait error");
pr_exit(status);           // & print its status

if((pid=fork())<0)
    printf("fork error");
else if(pid==0)    //child
    abort();        // generates SIGABRT

if(wait(&status) != pid)    //wait for child
    printf("wait error");
pr_exit(status);

if((pid=fork())<0)
    printf("fork error");
else if(pid==0)    //child
    status/=0;      // divide by 0 generates SIGFPE

if(wait(&status) != pid)    //wait for child
    printf("wait error");
pr_exit(status);

return 0;
}
output:

```

```
modern@lab3-005:~/AOS
File Edit View Search Terminal Help
[modern@lab3-005 AOS]$ vi aossli20-1.c
[modern@lab3-005 AOS]$ gcc aossli20-1.c
aossli20-1.c: In function 'main':
aossli20-1.c:53:9: warning: division by zero [-Wdiv-by-zero]
    status/=0; // divide by 0 generates SIGFPE
    ^
[modern@lab3-005 AOS]$ ./a.out

normal termination
exit status = 7

abnormal termination
signal number = 6(Core file generated)

abnormal termination
signal number = 8(Core file generated)
[modern@lab3-005 AOS]$
```

/*Write a C program which blocks SIGQUIT signal for 5 seconds. After 5 second process checks any occurrence of quit signal during this period, if so, it unblock the signal. Now another occurrence of quit signal terminates the program. (Use sigprocmask() and sigpending())*/

Solution →

```
#include<stdio.h>
#include<signal.h>
static void sig_quit(int signo)
{
    printf("\nCaught SIG_QUIT");
    if(signal(SIGQUIT,SIG_DFL)==SIG_ERR)
        printf("\ncan't reset SIGQUIT");
}
int main()
{
    sigset_t newmask, oldmask, pendmask;
    if(signal(SIGQUIT,sig_quit)==SIG_ERR)
        printf("\ncant catch sigquit");
    sigemptyset(&newmask);
    sigaddset(&newmask, SIGQUIT);
    if(sigprocmask(SIG_BLOCK, &newmask, &oldmask)<0)
        printf("\nsigblock error");
    sleep(5);
    printf("old signal set : %8ld.\n",oldmask);
    if(sigpending(&pendmask)<0)
```

```

        printf("\nsig-pending error");
printf("pending signal set : %8.8ld.\n",pendmask);
if(sigismember(&pendmask,SIGQUIT))
    printf("\nSIGQUIT pending");
if(sigprocmask(SIG_SETMASK, &oldmask, NULL)<0)
    printf("\nsig_setmask error");

printf("\nSIGQUIT unblocked");
printf("\nhello\n");
sleep(10);
printf("\nhello\n");
return 0;
}

```

output:

```

modern@lab3-005:~/AOS
File Edit View Search Terminal Help
[modern@lab3-005 AOS]$ vi aosslipl-2.c
[modern@lab3-005 AOS]$ gcc aosslipl-2.c
[modern@lab3-005 AOS]$ ./a.out
old signal set : 140720339836400.
pending signal set : 00000008.

SIGQUIT unblocked
hello

hello
^C
[1]- Terminated ./a.out
[2]+ Terminated ./a.out
[modern@lab3-005 AOS]$ █

```


/*Write a C program to demonstrates the different behavior that can be seen with automatic, global, register, static and volatile variables (Use setjmp() and longjmp() system call).*/

Solution →

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

static void f1(int,int,int,int);
static void f2(void);

static jmp_buf jmpbuffer;
int globval;

int main()
{
    int autoval;
    register int regval;
    volatile int volval;
    static int statval;

    globval=1;autoval=2;regval=3;volval=4;statval=5;

    if(setjmp(jmpbuffer)!=0)
    {
        printf("\nAfter longjmp:\n");
        printf("\nGlobal value = %d\n",globval);
        printf("\nAuto value = %d\n",autoval);
        printf("\nRegister value = %d\n",regval);
        printf("\nVolatile value = %d\n",volval);
        printf("\nStatic value = %d\n",statval);

        exit(0);
    }

    globval=95;autoval=96;regval=97;volval=98;statval=99;
    f1(autoval,regval,volval,statval);

    return 0;
}

static void f1(int i,int j,int k,int l)
{
    printf("\nIn f1():\n");
    printf("\nGlobal value = %d\n",globval);
```

```

        printf("\nAuto value = %d\n",i);
        printf("\nRegister value = %d\n",j);
        printf("\nVolatile value = %d\n",k);
        printf("\nStatic value = %d\n",l);
        f2();
    }

```

```

static void f2(void)
{
    longjmp(jmpbuffer,1);
}

```

output:

```

[modern@lab3-005 AOS]$ vi aossli2-2.c
[modern@lab3-005 AOS]$ gcc aossli2-2.c
[modern@lab3-005 AOS]$ ./a.out

```

In f1():

Global value = 95

Auto value = 96

Register value = 97

Volatile value = 98

Static value = 99

After longjmp:

Global value = 95

Auto value = 96

Register value = 97

Volatile value = 98

Static value = 99

[modern@lab3-005 AOS]\$ █

/* Write a C program to implement the following unix/linux command (use fork, pipe and exec system call). Your program should block the signal Ctrl-C and Ctrl-\ signal during the execution. ls | */

Solution →

```

#include<stdio.h>
#include<unistd.h>
#include<signal.h>

```

```

static void sig_handler(int signo)
{
    if(signo == SIGINT)
        printf("\nCaught SIG_INT");
    if(signo == SIGQUIT)
        printf("\nCaught SIG_INT");
    if(signal(SIGINT,SIG_DFL)==SIG_ERR)
        printf("\ncan't reset SIGINT");
    if(signal(SIGQUIT,SIG_DFL)==SIG_ERR)
        printf("\ncan't reset SIGQUIT");
}

int main()
{
    sigset_t newmask, oldmask, pendmask;

    if(signal(SIGINT,sig_handler)==SIG_ERR)
        printf("\ncant catch sigint");
    if(signal(SIGQUIT,sig_handler)==SIG_ERR)
        printf("\ncant catch sigquit");
    sigemptyset(&newmask);
    sigaddset(&newmask, SIGINT);
    sigaddset(&newmask, SIGQUIT);
    int pipefd[2],retstatus,pid;
    retstatus = pipe(pipefd);
    if(retstatus==-1)
    {
        printf("\nfailed to create pipe\n");
        return 1;
    }
    pid=fork();
    if(pid<0)
    {
        printf("\nfailed to create child\n");
        return 2;
    }
    else if(pid==0) //child process
    {
        close(pipefd[0]);    //close read end of pipe
        close(1);           // close std.output
        dup(pipefd[1]);
        execlp("ls","ls","-l",(char *)0);
    }
    else                // parent process
    {

```

```

        close(pipefd[1]);    // close write end of pipe
        close(0);           // close std. input
        dup(pipefd[0]);
        if(sigprocmask(SIG_BLOCK, &newmask, &oldmask)<0)
            printf("\nsigblock error");
        sleep(5);

        if(sigpending(&pendmask)<0)
            printf("\nsig-pending error");

        if(sigismember(&pendmask,SIGINT))
            printf("\nSIGINT pending");

        if(sigismember(&pendmask,SIGQUIT))
            printf("\nSIGQUIT pending");

        if(sigprocmask(SIG_SETMASK, &oldmask, NULL)<0)
            printf("\nsig_setmask error");

        printf("\nSIGINT unblocked");
        printf("\nSIGQUIT unblocked");
        sleep(10);
        execlp("wc","wc","-l",(char *)0);

    }
    return 0;
}

```

output:

```

modern@lab3-005:~/AOS
File Edit View Search Terminal Help
[modern@lab3-005 AOS]$ vi aossli4-2.c
[modern@lab3-005 AOS]$ gcc aossli4-2.c
[modern@lab3-005 AOS]$ ./a.out
^C

SIGINT pending
Caught SIG_INT
SIGINT unblocked
57
[modern@lab3-005 AOS]$
[modern@lab3-005 AOS]$ █

```

/* Write a C program that behaves like a shell (command interpreter). It has its own prompt say . Any normal shell command is executed from your shell by starting a child process to execute the system program corresponding to the command. It should additionally interpret the following command.

i) list f<dirname> - print name of all files in directory

ii) list n <dirname> - print number of all entries

iii) list i<dirname> - print name and inode of all files

*/

Solution →

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

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

```
#include<unistd.h>
```

```
#include<string.h>
```

```
#include<sys/wait.h>
```

```
#include<dirent.h>
```

```
void separate_tokens(char *cmd,char *tok[])
```

```
{
```

```
    int i=0;
```

```
    char *p;
```

```
    p=strtok(cmd," ");
```

```
    puts(p);
```

```
    while(p!=NULL)
```

```
    {
```

```
        tok[i++]=p;
```

```
        p=strtok(NULL," ");
```

```
    }
```

```
    tok[i]=NULL;
```

```
}
```

```
void list(char *dirName,char param)
```

```
{
```

```
    DIR *dir;
```

```
    int count=0;
```

```
    struct dirent *entry;
```

```
//    struct stat buff;
```

```
    if((dir=opendir(dirName))==NULL)
```

```
    {
```

```
        printf("\n\tDirectory %s notfound\n",dirName);
```

```
        return;
```

```
    }
```

```
    switch(param)
```

```
    {
```

```
        case 'f':        while((entry=readdir(dir))!=NULL)
```

```

                                printf("\n%s",entry->d_name);
                                break;
case 'n':    while((entry=readdir(dir))!=NULL)
                                count++;
                                printf("\nTotal number of entries = %d\n",count);
                                break;
case 'i':while((entry=readdir(dir))!=NULL)
                                printf("\n%ld:%s",entry->d_ino,entry-
>d_name);
                                break;
    }
}

int main()
{
    char cmd[80],*args[10];
    int pid;

    system("clear");
    do
    {
        printf("\nNewShell$ ");
        fgets(cmd,80,stdin);
        cmd[strlen(cmd)-1]='\0';
        separate_tokens(cmd,args);

        if(strcmp(args[0],"list")==0)
            list(args[2],args[1][0]);
        else
        {
            pid = fork();
            if(pid > 0)
                wait(0);
            else if(execvp(args[0],args)==-1)
                printf("\n Command %s not found\n",args[0]);
        }
    }while(1);
    return 0;
}

```

output:

```

NewShell$ list f modern
list

.
..
p1.txt
p2.txt
NewShell$ list n modern
list

Total number of entries = 4

NewShell$ list i modern
list

100761089:..
100761057:..
100765279:p1.txt
100765283:p2.txt
NewShell$ ^Z
[3]+  Stopped                  ./a.out
[modern@lab3-005 AOS]$ █

```

/* Write a program to file and directories information*/
Solution →

```

#include<sys/stat.h>
#include<string.h>
#include<stdio.h>
#include<stdlib.h>
#include<time.h>
#include<sys/types.h>
#include<pwd.h>
#include<grp.h>
#include<dirent.h>
#include<unistd.h>

int main(int argc,char *argv[])
{
    DIR *dirptr;
    struct dirent *entry;
    char curDir[80];
    getcwd(curDir,80);
    printf("%s\n",curDir);
    struct stat s;
    struct tm *timeinfo;
    struct passwd *pw;

```

```

struct group*gr;
char filetype,perm,*date;
memset(&s,0,sizeof(s));

dirptr = opendir(curDir);
while((entry = readdir(dirptr))!=NULL)
{
    printf("\n");
    stat(entry->d_name,&s);
    if((s.st_mode & S_IFMT)==S_IFREG) filetype = '-';
    else if((s.st_mode & S_IFMT)==S_IFSOCK) filetype = 'S';
    else if((s.st_mode & S_IFMT)==S_IFLNK) filetype = 'L';
    else if((s.st_mode & S_IFMT)==S_IFBLK) filetype = 'B';
    else if((s.st_mode & S_IFMT)==S_IFDIR) filetype = 'D';
    else if((s.st_mode & S_IFMT)==S_IFCHR) filetype = 'C';
    else if((s.st_mode & S_IFMT)==S_IFIFO) filetype = 'F';

    date = ctime(&s.st_atime);
    timeinfo = localtime(&s.st_atime);
    date[strlen(date)-1]='\0';
    pw = getpwuid(s.st_uid);
    gr = getgrgid(s.st_gid);

    printf("%c",filetype);
    printf((s.st_mode & S_IRUSR)?"r":"-");
    printf((s.st_mode & S_IWUSR)?"w":"-");
    printf((s.st_mode & S_IXUSR)?"x":"-");
    printf((s.st_mode & S_IRGRP)?"r":"-");
    printf((s.st_mode & S_IWGRP)?"w":"-");
    printf((s.st_mode & S_IXGRP)?"x":"-");
    printf((s.st_mode & S_IROTH)?"r":"-");
    printf((s.st_mode & S_IWOTH)?"w":"-");
    printf((s.st_mode & S_IXOTH)?"x":"-");
    printf(" %ld %s %s %ld\t%s %s",s.st_nlink,pw->pw_name,pw-
>pw_name,s.st_size,date,entry->d_name);

    }
    return 0;
}
output:

```



```
[modern@lab3-005 AOS]$ ./a.out  
/home/modern/AOS
```

```
Drwxr-xr-x 4 modern modern 4096 Wed Apr 26 15:04:23 2023 .  
Drwx----- 18 modern modern 4096 Wed Apr 26 14:59:38 2023 ..  
---x-----x 1 modern modern 21844109 Wed Apr 26 14:22:01 2023 file2.txt  
-rw-r--r-- 1 modern modern 1148 Wed Apr 26 14:28:23 2023 aosslip1-2.c  
-rw-rw-r-- 1 modern modern 36 Wed Apr 26 11:51:05 2023 dd.txt  
-rw-r--r-- 1 modern modern 401 Wed Apr 26 14:18:32 2023 aosslip16-1.c  
-rw-r--r-- 1 modern modern 2010 Wed Apr 26 11:57:07 2023 aosslip4-1.c  
-rw-r--r-- 1 modern modern 723 Wed Apr 26 13:58:09 2023 aosslip12-1.c  
-rw-r--r-- 1 modern modern 489 Wed Apr 26 14:19:05 2023 aosslip17-1.c  
Drwxr-xr-x 2 modern modern 34 Wed Apr 26 15:00:07 2023 modern  
-----wx--- 1 modern modern 1044 Wed Apr 26 12:14:27 2023 slip7-1.txt  
-rwxrwxrwx 1 modern modern 1045 Wed Apr 26 12:24:00 2023 pl.txt  
-rw-r--r-- 1 modern modern 452 Wed Apr 26 12:22:39 2023 aosslip7-1.c  
-rw-r--r-- 1 modern modern 1377 Wed Apr 26 12:25:56 2023 aosslip8-1.c  
-rw-r--r-- 1 modern modern 1211 Wed Apr 26 12:33:24 2023 aos1slip10-1.c  
-rw-r--r-- 1 modern modern 1211 Wed Apr 26 13:47:36 2023 aossslip10-1.c  
-rw-r--r-- 1 modern modern 419 Wed Apr 26 14:11:40 2023 aosslip13-1.c  
-rw-r--r-- 1 modern modern 626 Wed Apr 26 14:20:42 2023 aosslip18-1.c  
-rw-r--r-- 1 modern modern 1647 Wed Apr 26 15:04:19 2023 aosslip7-2.c  
-rw-r--r-- 1 modern modern 527686 Wed Apr 26 15:03:14 2023 FINALAOS.docx  
Drwxr-xr-x 3 modern modern 46 Wed Apr 26 11:29:23 2023 New folder  
-rw-r--r-- 1 modern modern 1131 Wed Apr 26 13:47:09 2023 aos10.c  
-rw-r--r-- 1 modern modern 1206 Wed Apr 26 13:51:54 2023 pl.c  
-rw-r--r-- 1 modern modern 301 Wed Apr 26 13:54:52 2023 aosslip11-1.c  
-rw-r--r-- 1 modern modern 1249 Wed Apr 26 14:27:06 2023 aosslip20-1.c  
-rw-r--r-- 1 modern modern 89 Wed Apr 26 15:03:14 2023 ~/.lock.FINALAOS.docx#  
-rw-r--r-- 1 modern modern 613 Wed Apr 26 14:12:54 2023 aosslip14-1.c  
-rw-r--r-- 1 modern modern 949 Wed Apr 26 14:14:00 2023 aosslip15-1.c  
-rw-r--r-- 1 modern modern 403 Wed Apr 26 14:25:55 2023 aosslip19-1.c  
-rw-r--r-- 1 modern modern 700 Wed Apr 26 11:35:25 2023 aosslip1-1.c  
-rw-rw-r-- 1 modern modern 23 Wed Apr 26 11:35:39 2023 mod.txt  
-rw-r--r-- 1 modern modern 1195 Wed Apr 26 14:31:13 2023 aosslip2-2.c
```

/* Write a C program which receives file names as command line arguments and display those filenames in ascending order according to their sizes.

(e.g \$ a.out a.txt b.txt c.txt, ...) */

Solution →

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

struct fileinfo
{
    char fileName[20];
    int size;
}files[20],temp;

int main(int argc,char *argv[])
{
    struct stat s;
    memset(&s,0,sizeof(s));
    int i,j,n;

    for(i=1;i<argc;i++)
    {
        printf("\n");
        stat(argv[i],&s);
        strcpy(files[i-1].fileName,argv[i]);
        files[i-1].size = s.st_size;
    }
    n=i-1;
    for(i=0;i<n;i++)
    {
        for(j=i+1;j<n;j++)
        {
            if(files[i].size > files[j].size)
            {
                temp = files[i];
                files[i]=files[j];
                files[j]=temp;
            }
        }
    }
}
```

```

    }
    for(i=0;i<n;i++)
        printf("\n%s\t%d",files[i].fileName,files[i].size);
    return 0;
}

```

output:

```

File Edit View Search Terminal Help
[modern@lab3-005 AOS]$ vi a.txt
[modern@lab3-005 AOS]$ vi z.txt
[modern@lab3-005 AOS]$ vi p.txt
[modern@lab3-005 AOS]$ vi aossli8-2.c
[modern@lab3-005 AOS]$ gcc aossli8-2.c
[modern@lab3-005 AOS]$ ./a.out a.txt z.txt p.txt

```

```

p.txt    7
a.txt    15
z.txt    16[modern@lab3-005 AOS]$ █

```

/* Write a C program that will only list all subdirectories in alphabetical order from current directory. */

Solution →

```

#include<sys/stat.h>
#include<string.h>
#include<stdio.h>
#include<stdlib.h>
#include<time.h>
#include<sys/types.h>
#include<pwd.h>
#include<grp.h>
#include<dirent.h>
#include<unistd.h>

int main(int argc,char *argv[])
{
    DIR *dirptr;
    struct dirent *entry;
    char curDir[80];
    getcwd(curDir,80);
    printf("%s\n",curDir);
    struct stat s;

```

```

char *files[10],temp[10];
int i=0,j,n;
memset(&s,0,sizeof(s));

dirptr = opendir(curDir);
while((entry = readdir(dirptr))!=NULL)
{
    stat(entry->d_name,&s);
    if((s.st_mode & S_IFMT)==S_IFDIR)
    {
        files[i]=malloc(20);
        strcpy(files[i++],entry->d_name);
    }
}
n=i-1;
for(i=0;i<n;i++)
{
    for(j=i+1;j<n;j++)
    {
        if(strcmp(files[i],files[j])>0)
        {
            strcpy(temp,files[i]);
            strcpy(files[i],files[j]);
            strcpy(files[j],temp);
        }
    }
}
for(i=0;i<n;i++)
    printf("\n%s",files[i]);

return 0;
}

```

output:

```

[modern@lab3-005 AOS]$ vi aossli9-2.c
[modern@lab3-005 AOS]$ ./a.out
/home/modern/AOS

.
..
abc
anaita
modern[modern@lab3-005 AOS]$ █

```

/* Write a C program to display all the files from current directory and its subdirectory whose size is greater than Bytes Where n is accepted from user through command line.
*/

Solution →

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

int main(int argc,char *argv[])
{
    DIR *dirptr,*subdirptr;
    struct dirent *entry,*subentry;
    struct stat s;
    memset(&s,0,sizeof(s));

    int n = atoi(argv[1]);
    char curDir[80];
    getcwd(curDir,80);
    if(argc<2)
    {
        printf("\n Insufficient arguments\n");
        exit(1);
    }
    dirptr = opendir(curDir);
    while((entry = readdir(dirptr))!=NULL)
    {
        stat(entry->d_name,&s);
        printf("\n%s",entry->d_name);
        if(((s.st_mode & S_IFMT)==S_IFREG) && s.st_size > n)
            printf("\n%s : %ld",entry->d_name,s.st_size);
        if((s.st_mode & S_IFMT)==S_IFDIR)
        {
            subdirptr = opendir(entry->d_name);
            while((subentry = readdir(subdirptr))!=NULL)
            {
                stat(subentry->d_name,&s);
                if(((s.st_mode & S_IFMT)==S_IFREG) && s.st_size > n)
                    printf("\n%s : %ld",subentry->d_name,s.st_size);
            }
        }
    }
}
```

```

    }
}
closedir(dirptr);
return 0;
}

```

output:

```

modern@lab3-005:~/AOS/abc
File Edit View Search Terminal Help
[modern@lab3-005 abc]$ gcc aosslipl2-2.c
[modern@lab3-005 abc]$ ./a.out 10000

.
..
p.txt
q.txt
aosslipl2-2.c
a.out[modern@lab3-005 abc]$ █

```

/* Write a C program that behaves like a shell (command interpreter). It has its own prompt say . Any normal shell command is executed from your shell by starting a child process to execute the system program corresponding to the command. It should additionally interpret the following command.

- i) typeline +10 <filename> - print first 10 lines of file
 - ii) typeline -20 <filename> - print last 20 lines of file
 - iii) typeline a <filename> - print all lines of file
- */

Solution →

```

#include<stdio.h>
#include<stdlib.h>
#include<unistd.h>
#include<string.h>
#include<sys/wait.h>
#include<dirent.h>
#include<sys/types.h>
#include<sys/stat.h>
#include<fcntl.h>

```

```

void separate_tokens(char *cmd,char *tok[])
{
    int i=0;
    char *p;

```

```

    p=strtok(cmd," ");
    puts(p);
    while(p!=NULL)
    {
        tok[i++]=p;
        p=strtok(NULL," ");
    }
    tok[i]=NULL;
}

void typeline(char *fileName,char* count)
{
    int handle,n,i=0,cnt=0;
    char ch;

    if((handle=open(fileName,O_RDONLY))== -1)
    {
        printf("\n\file %s notfound\n",fileName);
        return;
    }
    if(strcmp(count,"a")==0)
    {
        while((read(handle,&ch,1)!=0))
            printf("%c",ch);
        close(handle);
        return;
    }
    n=atoi(count);
    if(n>0)
    {
        while((read(handle,&ch,1)!=0))
        {
            if(ch=='\n') i++;
            if(i==n) break;
            printf("%c",ch);
        }
        printf("\n");
        close(handle);
        return;
    }
    if(n<0)
    {
        while((read(handle,&ch,1)!=0))
            if(ch=='\n') cnt++;
        lseek(handle,0,SEEK_SET);
        while((read(handle,&ch,1)!=0))
        {

```

```

        if(ch=='\n') i++;
        if(i==(cnt+n))
            break;
    }
    while((read(handle,&ch,1)!=0))
        printf("%c",ch);

    printf("\n");
    close(handle);
    return;
}
}

int main()
{
    char cmd[80],*args[10];
    int pid;

    system("clear");
    do
    {
        printf("\nNewShell$ ");
        fgets(cmd,80,stdin);
        cmd[strlen(cmd)-1]='\0';
        separate_tokens(cmd,args);

        if(strcmp(args[0],"typeline")==0)
            typeline(args[2],args[1]);
        else
        {
            pid = fork();
            if(pid > 0)
                wait(0);
            else if(execvp(args[0],args)==-1)
                printf("\n Command %s not found\n",args[0]);
        }
    }while(1);
    return 0;
}

```

Output:


```

NewShell$ typeline +10 a.txt
typeline
This is Modern
This is Pen.
This is Pencil.
This is purse.
This is keyboard.
This is Paper.
This is A.C.
This is Camera.
This is Society.
This is File.

NewShell$ typeline -20 a.txt
typeline
This is A.C.
This is Camera.
This is Society.
This is File.
This is Blue color.
This is Yellow color.
This is Black color.
This is Red color.
This is White color
This is pink color.
This is green color.
This is classroom.
This is table.
This is chair.
This is File.
This is a number.
This is CPU.

```

```

NewShell$ typeline a a.txt
typeline
This is Modern
This is Pen.
This is Pencil.
This is purse.
This is keyboard.
This is Paper.
This is A.C.
This is Camera.
This is Society.
This is File.
This is Blue color.
This is Yellow color.
This is Black color.
This is Red color.
This is White color
This is pink color.
This is green color.
This is classroom.
This is table.
This is chair.
This is File.
This is a number.
This is CPU.

```

/* Write a C program which creates a child process and child process catches a signal SIGHUP, SIGINT and SIGQUIT. The Parent process send a SIGHUP or SIGINT signal after every 3 seconds, at the end of 15 second parent send SIGQUIT signal to child and child terminates by displaying message "My Papa has Killed me!!!". */

Solution →

```

#include<stdio.h>
#include<signal.h>

```

```

#include<unistd.h>
#include<stdlib.h>

void sighup(int signo)
{
    signal(SIGHUP,sighup);
    printf("\nCHILD : I have received SIGHUP");
}

void sigint(int signo)
{
    signal(SIGINT,sigint);
    printf("\nCHILD : I have received SIGINT");
}

void sigquit(int signo)
{
    // signal(SIGQUIT,sigquit);
    printf("\nCHILD : My daddy has killed me");
    exit(0);
}

int main()
{
    int pid;
    struct sigaction sigact;
    sigact.sa_flags=0;
    sigemptyset(&sigact.sa_mask);

    sigact.sa_handler = sighup;
    if(sigaction(SIGHUP,&sigact,NULL)<0)
    {
        printf("\nsigaction error");
        exit(1);
    }

    sigact.sa_handler = sigint;
    if(sigaction(SIGINT,&sigact,NULL)<0)
    {
        printf("\nsigaction error");
        exit(1);
    }

    sigact.sa_handler = sigquit;
    if(sigaction(SIGQUIT,&sigact,NULL)<0)
    {
        printf("\nsigaction error");
    }
}

```

```

        exit(1);
    }

    if((pid=fork()) < 0)
    {
        printf("\nfork error");
        exit(1);
    }
    if(pid == 0)    //child
    {
        for(;;) ;
    }
    else            //parent
    {
        sigact.sa_handler = SIG_DFL;

        sigaction(SIGHUP,&sigact,NULL);
        sigaction(SIGINT,&sigact,NULL);
        sigaction(SIGQUIT,&sigact,NULL);

        printf("\nparent sending SIGHUP");
        kill(pid,SIGHUP);
        sleep(3);
        printf("\nparent sending SIGINT");
        kill(pid,SIGINT);
        sleep(3);

        printf("\nparent sending SIGHUP");
        kill(pid,SIGHUP);
        sleep(3);
        printf("\nparent sending SIGINT");
        kill(pid,SIGINT);
        sleep(3);

        printf("\nparent sending SIGINT");
        kill(pid,SIGINT);
        sleep(3);

        printf("\nparent sending SIGQUIT");
        kill(pid,SIGQUIT);
        sleep(3);
    }
    return 0;
}
output:

```

```
modern@lab3-005:~/AOS
File Edit View Search Terminal Help
[modern@lab3-005 AOS]$ gcc aossli15-2.c
[modern@lab3-005 AOS]$ vi aossli15-2.c
[modern@lab3-005 AOS]$ ./a.out

parent sending SIGHUP
CHILD : I have received SIGHUP
parent sending SIGINT
CHILD : I have received SIGINT
parent sending SIGHUP
CHILD : I have received SIGHUP
parent sending SIGINT
CHILD : I have received SIGINT
parent sending SIGINT
CHILD : I have received SIGINT
CHILD : My daddy has killed meparent sending SIGQUIT[modern@lab3-005 AOS]$
```

/*

Write a C program to implement the following unix/linux command on current directory

ls -l > output.txt

DO NOT simply exec ls -l > output.txt or system command from the program.

*/

Solution →

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

```
int main(int argc,char *argv[])
{
    DIR *dirptr;
    struct dirent *entry;
    char curDir[80];
    getcwd(curDir,80);
    printf("%s\n",curDir);
    struct stat s;
    struct tm *timeinfo;
    struct passwd *pw;
    struct group*gr;
    int fd;
    char filetype,perm,*date;
    memset(&s,0,sizeof(s));
```

```

fd=open("a.txt",O_CREAT | O_WRONLY);
chmod("a.txt",0777);
close(1);
dup(fd);

dirptr = opendir(curDir);
while((entry = readdir(dirptr))!=NULL)
{
    printf("\n");
    stat(entry->d_name,&s);
    if((s.st_mode & S_IFMT)==S_IFREG) filetype = '-';
    else if((s.st_mode & S_IFMT)==S_IFSOCK) filetype = 'S';
    else if((s.st_mode & S_IFMT)==S_IFLNK) filetype = 'L';
    else if((s.st_mode & S_IFMT)==S_IFBLK) filetype = 'B';
    else if((s.st_mode & S_IFMT)==S_IFDIR) filetype = 'D';
    else if((s.st_mode & S_IFMT)==S_IFCHR) filetype = 'C';
    else if((s.st_mode & S_IFMT)==S_IFIFO) filetype = 'F';

    date = ctime(&s.st_atime);
    timeinfo = localtime(&s.st_atime);
    date[strlen(date)-1]='\0';
    pw = getpwuid(s.st_uid);
    gr = getgrgid(s.st_gid);

    printf("%c",filetype);
    printf((s.st_mode & S_IRUSR)?"r":"-");
    printf((s.st_mode & S_IWUSR)?"w":"-");
    printf((s.st_mode & S_IXUSR)?"x":"-");
    printf((s.st_mode & S_IRGRP)?"r":"-");
    printf((s.st_mode & S_IWGRP)?"w":"-");
    printf((s.st_mode & S_IXGRP)?"x":"-");
    printf((s.st_mode & S_IROTH)?"r":"-");
    printf((s.st_mode & S_IWOTH)?"w":"-");
    printf((s.st_mode & S_IXOTH)?"x":"-");
    printf(" %ld %s %s %ld\t%s %s",s.st_nlink,pw->pw_name,pw-
>pw_name,s.st_size,date,entry->d_name);

    }
    close(fd);
    return 0;
}
output:

```

```

File Edit View Search Terminal Help
[modern@lab3-005 A05]$ wq
bash: wq: command not found...
[modern@lab3-005 A05]$
[modern@lab3-005 A05]$ vi aosslipl6-2.c
[modern@lab3-005 A05]$ gcc aosslipl6-2.c
[modern@lab3-005 A05]$ ./a.out
/home/modern/A05
[modern@lab3-005 A05]$ ls -l > a.txt
[modern@lab3-005 A05]$ cat a.txt
total 984
drwxr-xr-x. 2 modern modern    66 Apr 26 15:27 abc
-rwxrwxrwx. 1 modern modern  1045 Apr 26 12:20 abc.txt
-rw-rw-r--. 1 modern modern    12 Apr 26 12:15 abx.txt
drwxr-xr-x. 3 modern modern    46 Apr 22 15:58 anaita
-rw-r--r--. 1 modern modern  1131 Apr 26 13:47 aos10.c
-rw-r--r--. 1 modern modern  1211 Apr 22 15:44 aos1slip10-1.c
-rw-r--r--. 1 modern modern    21 May 23 2022 aosslipl0-1.txt
-rw-rw-r--. 1 modern modern     0 Apr 26 15:18 aosslipl0-2.c
-rw-rw-r--. 1 modern modern   301 Apr 26 13:54 aosslipl1-1.c
-rw-rw-r--. 1 modern modern  1204 Apr 26 15:22 aosslipl1-2.c
-rw-rw-r--. 1 modern modern   700 Apr 26 11:35 aosslipl1-1.c
-rw-rw-r--. 1 modern modern   723 Apr 26 13:58 aosslipl2-1.c
-rw-rw-r--. 1 modern modern  1148 Apr 26 14:28 aosslipl1-2.c
-rw-rw-r--. 1 modern modern   419 Apr 26 14:05 aosslipl3-1.c
-rw-rw-r--. 1 modern modern  2017 Apr 26 15:34 aosslipl3-2.c
-rw-rw-r--. 1 modern modern   613 Apr 26 14:12 aosslipl4-1.c
-rw-rw-r--. 1 modern modern   949 Apr 26 14:13 aosslipl5-1.c
-rw-rw-r--. 1 modern modern  1856 Apr 26 15:37 aosslipl5-2.c
-rw-rw-r--. 1 modern modern   401 Apr 26 14:18 aosslipl6-1.c
-- -- -- -- --

```

/*

Write a C program which creates a child process to run linux/ unix command or any user defined program. The parent process set the signal handler for death of child signal and Alarm signal. If a child process does not complete its execution in 5 second then parent process kills child process.

*/

Solution →

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

```
#include<unistd.h>
```

```
#include<sys/wait.h>
```

```
#include<signal.h>
```

```
pid_t pid;
```

```
static void sig_handler(int signo)
```

```
{
```

```
// if(signo==SIG_ERR)
```

```
// printf("\n sig err");
```

```
if(signo == SIGCHLD)
```

```
printf("\nchild signal");
```

```
if(signo == SIGALRM)
```

```
{
```

```
printf("\n alarm signal");
```

```
kill(pid,SIGKILL);
```

```
}
```

```
}
```

```
int main()
```

```
{
```

```
signal(SIGCHLD,sig_handler);
```

```
signal(SIGALRM,sig_handler);
```

```
if((pid=fork())<0)
```

```
printf("\nfork error");
```

```
if(pid==0)
```

```

{
    sleep(5);
    execlp("ls","ls","-l",NULL);
}
alarm(5);
wait(NULL);
return 0;
}

```

output:

```

[modern@lab3-005 AOS]$ gcc aosslipl9-2.c
[modern@lab3-005 AOS]$ ./a.out

```

```

alarm signal
child signal[modern@lab3-005 AOS]$ █

```

```

[modern@lab3-005 AOS]$ gcc aosslipl9-2.c
[modern@lab3-005 AOS]$ ./a.out
total 1068
drwxr-xr-x. 2 modern modern    66 Apr 26 15:27 abc
-rwxrwxrwx. 1 modern modern  1045 Apr 26 12:20 abc.txt
-rw-rw-r--. 1 modern modern    12 Apr 26 12:15 abx.txt
drwxr-xr-x. 3 modern modern    46 Apr 22 15:58 anaita
-rw-r--r--. 1 modern modern  1131 Apr 26 13:47 aos10.c
-rw-r--r--. 1 modern modern  1211 Apr 22 15:44 aos1slip10-1.c
-rw-r--r--. 1 modern modern    21 May 23 2022 aosslipl0-1.txt
-rw-rw-r--. 1 modern modern     0 Apr 26 15:18 aosslipl0-2.c
-rw-r--r--. 1 modern modern   301 Apr 26 13:54 aosslipl1-1.c
-rw-r--r--. 1 modern modern  1204 Apr 26 15:22 aosslipl1-2.c
-rw-r--r--. 1 modern modern   700 Apr 26 11:35 aosslipl1-1.c
-rw-r--r--. 1 modern modern   723 Apr 26 13:58 aosslipl2-1.c
-rw-r--r--. 1 modern modern  1148 Apr 26 14:28 aosslipl1-2.c
-rw-r--r--. 1 modern modern   419 Apr 26 14:05 aosslipl3-1.c
-rw-r--r--. 1 modern modern  2017 Apr 26 15:34 aosslipl3-2.c
-rw-r--r--. 1 modern modern   613 Apr 26 14:12 aosslipl4-1.c
-rw-r--r--. 1 modern modern   949 Apr 26 14:13 aosslipl5-1.c
-rw-r--r--. 1 modern modern  1856 Apr 26 15:37 aosslipl5-2.c
-rw-r--r--. 1 modern modern   401 Apr 26 14:18 aosslipl6-1.c

```