**#tw1**

#define \_POSIX\_SOURCE

#define \_POSIX\_C\_SOURCE 199309L

#include<iostream.h>

#include<limits.h>

#include<unistd.h>

using namespace std;

int main(){

int ch;

int res;

cout<<"Enter the choice: ";

cin>> ch;

switch(ch){

case 1:

cout <<"Run time configuration limits: ";

if((res = sysconf(\_SC\_CLK\_TCK)) == -1)

cout<<"System does not support\n";

else

cout<<"Number of clock ticks: "<<res<<endl;

if((res = sysconf(\_SC\_CHILD\_MAX)) == -1)

cout<<"System does not support\n";

else

cout<<"Maximum number of child process that process can create: "<<res<<endl;

if((res = pathconf("/",\_PC\_PATH\_MAX)) == -1)

cout<<"System does not support\n";

else

cout<<"Maximum path length: "<<res<<endl;

if((res = pathconf("/",\_PC\_NAME\_MAX)) == -1)

cout<<"System does not support\n";

else

cout<<"Maximum number of characters in a filename: "<<res<<endl;

if((res = sysconf(\_SC\_OPEN\_MAX)) == -1)

cout<<"System does not support\n";

else

cout<<"Maximum number of opend file per process: "<<res<<endl;

break;

case 2:

cout<<"Compile time configuration limits:";

cout<<"Max no of child process created by process"<<\_POSIX\_CHILD\_MAX<<endl;

cout<<"Max no of files opened by a process: "<<\_POSIX\_OPEN\_MAX<<endl;

cout<<"Max no of characters allowed in a path name:"<<\_POSIX\_PATH\_MAX<<endl;

cout<<"Max no of characters allowed in a file name: "<<\_POSIX\_NAME\_MAX<<endl;

}

return 0;

}

**#tw2**

#include <iostream>

#define \_POSIX\_SOURCE

#define \_POSIX\_C\_SOURCE 199309L

#include <unistd.h>

using namespace std;

int main()

{

#ifdef \_POSIX\_JOB\_CONTROL

cout << "System supports job control\n";

#else

cout << "System does not support job control\n";

#endif

#ifdef \_POSIX\_SAVED\_IDS

cout << "System supports saved set-UID and saved

set-GID\n";

#else

cout << "System does not support saved set-UID

and saved set-GID\n";

#endif

#ifdef \_POSIX\_CHOWN\_RESTRICTED

cout << "chown restricted option is: " <<

\_POSIX\_CHOWN\_RESTRICTED <<endl;

#else

cout << "System does not support system-wide

chown\_restricted option\n";

#endif

#ifdef \_POSIX\_NO\_TRUNC

cout << "Pathname trucnation option is: " <<

\_POSIX\_NO\_TRUNC << endl;

#else

cout << "System does not support system-wide

pathname trucnation option\n";

#endif

#ifdef \_POSIX\_VDISABLE

cout << "Diable character for terminal files is: "

<< \_POSIX\_VDISABLE << endl;

#else

cout<<"System does not support

\_POSIX\_VDISABLE\n";

#endif

return 0;

}

**#tw3**

#include<iostream>

#include<sys/types.h>

#include<unistd.h>

#include<string.h>

using namespace std;

int main(int argc,char\* argv[]){

if(((argc < 3) || (argc>4)) || (argc == 4 && strcmp(argv[1], "-s"))){

cerr<<"Usage: "<<argv[0]<<" [-s] <orig\_file> <new\_file>\n"

return 1;

}

if(argc == 4){

if(symlink(argv[2],argv[3]) == -1){

perror ("link");

return 1;

}

cout<<"Symbolic | Soft link of file created successfully\n"

}

if(argc == 3){

if(link(argv[1],argv[2]) == -1){

perror ("link");

return 1;

}

cout<<"Hardlink of file created successfully\n"

}

return 0;

}

**#tw4**

#include<stdio.h>

#include<sys/types.h>

#include<fcntl.h>

#include<unistd.h>

int main(int argc, char\*argv[]){

char temp[1000];

setbuf(stdout,temp);

struct flock fvar;

int fdesc;

int buf;

int rc;

off\_t offset;

pid\_t pid = fork();

fdesc = open(argv[1],0, \_RDWR);

offset = lseek(fdesc, -100,SEEK\_END);

fvar.l\_type = F\_WRLCK;

fvar.l\_whence = SEEK\_CUR;

fvar.l\_start = 0;

fvar.l\_len = 100;

if(fcntl(fdesc, F\_SETLK, &fvar) == -1){

printf("\n........File has been locked by:\n");

while(fcntl(fdesc, F\_GETLK, &fvar) != -1 && fvar.l\_type != F\_UNLCK){

printf("File: %s is locked by process with pid : %u",argv[1],fvar.l\_pid);

printf("from %ld th byte in the file for %ld", fvar.l\_start,fvar.l\_len);

printf("number of bytes for %s\n",(fvar.l\_type == F\_WRLCK?"wirte":"read"));

if(!fvar.l\_len)

break;

fvar.l\_start += fvar.l\_len;

fvar.l\_len = 0;

}

}

else{

printf("\n......................\n");

printf("\n File: %s was not locked and acquiring of exlcusive lock was",argv[1]);

printf("successful by process ID : %u\n"getpid());

offset = lseek(fdesc,-50,SEEK\_END);

printf("\n\n Last 50 bytes of line : %s = \n",argv[1]);

while((rc = read(fdesc,&buf,1)) > 0) printf("%c",buf);

offset = lseek(fdesc, -100, SEEK\_END);

fvar.l\_type = F\_UNLCK:

fvar.l\_whence = SEEK\_CUR;

fvar.l\_start = 0;

fvar.l\_len =100;

fcntl(fdesc,F\_SETLKW, &fvar);

print("\n File unlocked successfully\n\n");

}

return 0;

}

**#tw5**

#include<stdio.h>

#include<unistd.h>

int main(){

pid\_t t;

t = fork();

if(t == 0){

printf("Child having id %d\n",getpid());

}

else{

printf("Parent having id %d\n", getpid());

sleep(15);

}

}

**#tw6**

#define \_POSIX\_SOURCE

#define \_POSIX\_C\_SOURCE 199309L

#include<iostream>

#include<stdio.h>

using namespace std;

static void charatatime(char\*);

int main(void){

pid\_t pid;

pid = fork();

if(pid < 0){

printf("fork error");

}

else if(pid == 0){

charatatime("output from child");

}

else{

charatatime("output from parent");

}

return 0;

}

static void charatatime(char \*str){

char \*ptr;

int c;

setbuf(stdout,NULL);

for(ptr = str; (c=\*ptr++)!=0 ; )

putc(c.stdout);

}

**#tw7 client**

#include<stdio.h>

#include<stdlib.h>

#include<string.h>

#include<sys/socket.h>

#include<sys/types.h>

#include<netinet/in.h>

#include<arpa/inet.h>

#define PORT 4444

int main(){

int sockfd;

struct sockaddr\_in servAddr;

char buffer[1024];

sockfd = socket(AF\_INET,SOCK\_STREAM,0);

printf("[+] Client socket created successfully\n");

bzero(&servAddr, sizeof(servAddr));

servAddr.sin\_family = AF\_INET;

servAddr.sin\_port = htons(PORT);

serAddr.sin\_addr.s\_addr = inet\_addr("127.0.0.1");

connect(sockfd,(struct sockaddr\*)&servAddr, sizeof(servAddr));

printf("[+] Connected to server\n");

recv(sockfd, buffer, 1024, 0);

printf("[+] Data received from server: %s\n",buffer);

printf("[+] Closing the connection\n");

return 0;

}

**#tw7 server**

#include<sys/socket.h>

#include<netinet/in.h>

#include<arpa/inet.h>

#include<stdio.h>

#include<stdlib.h>

#include<strings.h>

#include<string.h>

#define PORT 4444

int main(){

int listenfd, connfd;

struct sockAddr\_in servAddr, cliAddr;

socklen\_t clilen;

char buffer[1024];

listenfd = socket(AF\_INET,SOCK\_STREAM,0)

printf("[+] server socket created successfully");

bzero(&servAddr, sizeof(servAddr));

servAddr.sin\_family = AF\_INET;

servAddr.sin\_port = htons(PORT);

servAddr.sin\_addr.s\_addr = inet\_addr("127.0.0.1");

bind(listenfd,(struct sockAddr\*)&servAddr, sizeof(servAddr));

printf("[+] binded to port %d successfully",PORT);

listen(listenfd,5);

printf("Listening.....");

connfd = accept(listenfd,(struct sockAddr\*)&cliAddr, clilen);

printf("[+] accept incoming connections");

strcpy(buffer, "Hello, How are you");

send(connfd,buffer,strlen(buffer),0);

printf("[+] sending message");

printf("closing the connection");

return 0;

}

}