Slip1\_1.c

/\* 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>

int get\_inode (int fd)

{

struct stat buf;

int ret;

ret = fstat (fd, &buf);

if (ret < 0)

{

perror ("fstat");

//return −1;

}

return buf.st\_ino;

}

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

{

int fd, inode,i;

if (argc < 2)

{

fprintf (stderr, "usage: %s <file>\n", argv[0]);

//return 1;

}

for(i=0;i<argc;i++)

{

fd = open (argv[i], O\_RDONLY);

if (fd < 0)

{

perror ("open");

//return 1;

}

inode = get\_inode(fd);

printf ("%d\n", inode);

}

return 0;

}

Slip1\_2.c

Q.2) 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)

Answer:-

#include <fcntl.h>

#include <unistd.h>

#include <stdio.h>

#include<signal.h>

#include<sys/types.h>

#include<sys/wait.h>

#include <stdlib.h>

void Dingdong()

{

printf("Ding!");

exit(1);

}

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

{

int PauseSecond,i;

if(argc!=3){

printf("\n How much seconds you want to sleep the child process\n");

}

for(i=0;i<argc;i++){

PauseSecond=atoi((argv[i]));

if(fork()==0)

{

printf(" \n%d waiting for alarm to go off\n",getpid());

printf("%d second pause",PauseSecond);

printf("\n fgf");

printf("\n");

sleep(PauseSecond);

kill(getpid(),SIGALRM);

}

else {

printf("\n %d Alarm application starting\n", getpid());

signal(SIGALRM,Dingdong);

printf("\ndone");

}

}

}

Slip2

Q.1) 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.

Answer:-

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

int main(int argc,char\* argv[])  
{  
struct stat info;  
if(argc!=2)  
{  
printf("Enter a filename");  
scanf("%s",argv[1]);  
}  
if(stat(argv[1],&info)==-1)  
{  
printf("stat error/n");  
exit(0);  
}  
printf("inode number=%d\n",info.st\_ino);  
printf("size = %d",(long)info.st\_size);  
printf("last file access = %s\n",ctime(&info.st\_atime));  
printf("notification time = %s\n",ctime(&info.st\_mtime));  
printf("No of Hardlink = %d\n",info.st\_nlink);  
printf("File Permissions : \n");  
printf((info.st\_mode && S\_IRUSR)?"r":"-");  
printf((info.st\_mode && S\_IWUSR)?"w":"-");  
printf((info.st\_mode && S\_IXUSR)?"x":"-");

return 0;

}

Q.2) 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.

Answer:-

#include <stdio.h>  
#include <unistd.h>  
#include <stdlib.h>  
#include <signal.h>  
void sigfun(int sig)  
{  
printf("You have presses Ctrl-C , please press again to exit");  
(void) signal(SIGINT, SIG\_DFL);  
}  
int main()  
{  
(void) signal(SIGINT, sigfun);  
while(1) {  
printf("Hello World!  
");  
sleep(1);  
}  
return(0);  
}

Slip3:-

Q.1) Print the type of file and inode number where file name accepted through Command Line

#include<stdio.h>

#include<stdlib.h>

#include<fcntl.h>

#include<unistd.h>

#include<sys/stat.h>

#include<sys/types.h>

#include<dirent.h>

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

{

struct stat fileStat;

char fnm[30];

int fd=0;

FILE \*filename;

printf("Enter file name= ");

scanf("%s",fnm);

if ( ( fd = open (fnm , O\_RDONLY) ) == -1)

{

perror ( "open " );

system("pause");

exit (1) ;

}

if(fstat(fd, &fileStat)<0)

return 1;

printf("Information for %s\n",fnm); \_\_\_\_\_\_\_\_\_\_// expected filetype syntax here system("pause"); return 0;

}

Q.2) 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

Answer:-

#include <stdio.h>

#include <stdlib.h>

#include <unistd.h>

#include <sys/types.h>

#include <sys/wait.h>

#include <signal.h>

// Signal handler for child's death

void childHandler(int signum) {

if (signum == SIGCHLD) {

printf("Child process terminated.\n");

exit(0);

}

}

// Signal handler for alarm

void alarmHandler(int signum) {

if (signum == SIGALRM) {

printf("Child process exceeded time limit. Killing...\n");

kill(getpid(), SIGKILL);

}

}

int main() {

pid\_t child\_pid;

// Set signal handlers

signal(SIGCHLD, childHandler);

signal(SIGALRM, alarmHandler);

// Create a child process

child\_pid = fork();

if (child\_pid == -1) {

perror("fork");

exit(EXIT\_FAILURE);

}

if (child\_pid == 0) {

// Child process

printf("Child process executing...\n");

// Replace the following line with the command you want to execute

execlp("ls", "ls", "-l", NULL);

// If execlp fails

perror("execlp");

exit(EXIT\_FAILURE);

} else {

// Parent process

// Set an alarm for 5 seconds

alarm(5);

// Wait for the child process to complete

wait(NULL);

// Cancel the alarm

alarm(0);

printf("Parent process exiting...\n");

}

return 0;

}

Slip4 :

Q.1) Write a C program to find whether a given files passed through command line arguments are present in current directory or not.

Answer:-

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

int main(int argc,char \*argv[])  
{  
if(access(argv[1],F\_OK)==0)  
printf("File %s exists.",argv[1]);  
else  
printf("File not exists.");  
return 0;  
}

Q.2) 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!!!”

Answer:-

#include <signal.h>

#include <stdio.h>

#include <stdlib.h>

#include <sys/types.h>

#include <unistd.h>

// function declaration

void sighup();

void sigint();

void sigquit();

// driver code

void main()

{

    int pid;

    /\* get child process \*/

    if ((pid = fork()) < 0) {

        perror("fork");

        exit(1);

    }

    if (pid == 0) { /\* child \*/

        signal(SIGHUP, sighup);

        signal(SIGINT, sigint);

        signal(SIGQUIT, sigquit);

        for (;;)

            ; /\* loop for ever \*/

    }

    else /\* parent \*/

    { /\* pid hold id of child \*/

        printf("\nPARENT: sending SIGHUP\n\n");

        kill(pid, SIGHUP);

        sleep(3); /\* pause for 3 secs \*/

        printf("\nPARENT: sending SIGINT\n\n");

        kill(pid, SIGINT);

        sleep(3); /\* pause for 3 secs \*/

        printf("\nPARENT: sending SIGQUIT\n\n");

        kill(pid, SIGQUIT);

        sleep(3);

    }

}

// sighup() function definition

void sighup()

{

    signal(SIGHUP, sighup); /\* reset signal \*/

    printf("CHILD: I have received a SIGHUP\n");

}

// sigint() function definition

void sigint()

{

    signal(SIGINT, sigint); /\* reset signal \*/

    printf("CHILD: I have received a SIGINT\n");

}

// sigquit() function definition

void sigquit()

{

    printf("My DADDY has Killed me!!!\n");

    exit(0);

}

Slip5:

Q.1) Read the current directory and display the name of the files, no of files in current directory

Answer:-

#include<stdio.h>

#include<dirent.h>

int main(void)

{

DIR \*d;

struct dirent \*dir;

d = opendir(".");

if (d)

{

while ((dir = readdir(d)) != NULL)

{

printf("%s\n", dir->d\_name);

}

closedir(d);

}

return(0);

}

Q.2) 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 = “Hello World”  
Message2 = “Hello SPPU”  
Message3 = “Linux is Funny”

Answer:-

#include <stdio.h>

#include <unistd.h>

#include <sys/types.h>

#include <sys/wait.h>

#define BUFFER\_SIZE 100

int main() {

int pipefd[2];

char buffer[BUFFER\_SIZE];

if (pipe(pipefd) == -1) {

perror("Pipe creation failed");

return 1;

}

pid\_t pid = fork();

if (pid == -1) {

perror("Fork failed");

return 1;

} else if (pid == 0) {

// Child process

close(pipefd[0]); // Close the reading end of the pipe

// Write messages to the pipe

write(pipefd[1], "Hello World", 12);

write(pipefd[1], "Hello SPPU", 11);

write(pipefd[1], "Linux is Funny", 15);

close(pipefd[1]); // Close the writing end of the pipe

return 0;

} else {

// Parent process

close(pipefd[1]); // Close the writing end of the pipe

printf("Messages received from child process:\n");

// Read and display messages from the pipe

while (read(pipefd[0], buffer, BUFFER\_SIZE) > 0) {

printf("%s\n", buffer);

}

close(pipefd[0]); // Close the reading end of the pipe

wait(NULL); // Wait for the child process to terminate

return 0;

}

}

Slip6:

Q.1) Display all the files from current directory which are created in particular month

Answer:

#include<stdio.h>  
#include<dirent.h>  
#include<string.h>  
#include<sys/stat.h>  
#include<time.h>  
#include<stdlib.h>  
int main(intargc, char \*argv[])  
{  
char in[100],st[100],\*ch,\*ch1,c,buff[512];  
DIR \*dp;  
int i;  
structdirent \*ep;  
struct stat sb;  
charmon[100];  
dp=opendir("./");  
if (dp != NULL)  
{  
while(ep =readdir(dp))  
{  
if(stat(ep->d\_name,&sb) == -1)  
{  
perror("stat");  
exit(EXIT\_SUCCESS);  
}  
strcpy(mon,ctime(&sb.st\_ctime));  
ch=strtok(mon," ");  
ch=strtok(NULL,",");  
ch1=strtok(ch," ");  
if((strcmp(ch1,argv[1]))==0)  
{  
printf("%s\t\t%s",ep->d\_name,ctime(&sb.st\_ctime));  
}  
}  
(void)closedir(dp);  
}  
return 0;  
}

Q.2) Write a C program to create n child processes. When all n child processes terminates, Display total cumulative time children spent in user and kernel mode

Answer:

#include<sys/types.h>

#include<sys/wait.h>

#include<unistd.h>

#include<time.h>

#include<sys/times.h>

#include<stdio.h>

#include<stdlib.h>

int main(void)

{

int i, status;

 pid\_t pid;

 time\_t currentTime;

struct tms cpuTime;

if((pid = fork())==-1) //start child process

 {

 perror("\nfork error");

 exit(EXIT\_FAILURE);

 }

else if(pid==0) //child process

 {

 time(&currentTime);

 printf("\nChild process started at %s",ctime(&currentTime));

 for(i=0;i<5;i++)

 {

 printf("\nCounting= %dn",i); //count for 5 seconds

 sleep(1);

 }

 time(&currentTime);

 printf("\nChild process ended at %s",ctime(&currentTime));

 exit(EXIT\_SUCCESS);

 }

else

 { //Parent process

 time(&currentTime); // gives normal time

 printf("\nParent process started at %s ",ctime(&currentTime));

 if(wait(&status)== -1) //wait for child process

 perror("\n wait error");

 if(WIFEXITED(status))

 printf("\nChild process ended normally");

 else

 printf("\nChild process did not end normally");

 if(times(&cpuTime)<0) //Get process time

 perror("\nTimes error");

 else

 { // \_SC\_CLK\_TCK: system configuration time: seconds clock tick

 printf("\nParent process user time= %fn",((double)

cpuTime.tms\_utime));

 printf("\nParent process system time = %fn",((double)

cpuTime.tms\_stime));

printf("\nChild process user time = %fn",((double)

cpuTime.tms\_cutime));

 printf("\nChild process system time = %fn",((double)

cpuTime.tms\_cstime));

 }

 time(&currentTime);

 printf("\nParent process ended at %s",ctime(&currentTime));

 exit(EXIT\_SUCCESS);

 }

}

Slip7

Q.1) Write a C Program that demonstrates redirection of standard output to a file

Example:- ls>f1.

Answer

#include<stdlib.h>

#include<stdio.h>

#include<string.h>

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

{

char d[50];

if(argc==2)

{

bzero(d,sizeof(d));

strcat(d,"ls ");

strcat(d,"> ");

strcat(d,argv[1]);

system(d);

}

else

printf("\nInvalid No. of inputs");

}

output:

student@ubuntu:~$ gcc –o std.out std.c

student@ubuntu:~$ls

downloads documents listing.c listing.out std.c std.out

student@ubuntu:~$ cat > f1

^z

student@ubuntu:~$./std.out f1

student@ubuntu:~$cat f1

downloads

documents

listing.c

listing.out

std.c

std.out

Q.2 Implement the following unix/linux command (use fork, pipe and exec system call) ls –l | wc –l

Answer:-

#include<stdio.h>

#include<unistd.h>

int main()

{

int fd[2],dupFd;

char \*filename1 ="ls";

char \*filename2 ="wc";

char \*arg1 = "-l";

pipe(fd);

if(!fork())// return 0 for child process and 1 for parent process

{

close(1); // 1 for closing stdout

dup(fd[1]);

close(fd[0]);

execlp(filename1,filename1,arg1,NULL);

}else

{

close(0);

dup(fd[0]);

close(fd[1]);

execlp(filename2,filename2,arg1,NULL);

}

}

Output is

(base) amol@amol-Ideapad-320:~/AOS$ ./a.out

total 140

drwxrwxr-x 2 amol amol 4096 May 28 16:19 adir

-rw-rw-r-- 1 amol amol 11962 May 29 13:59 AOSSolution2022.txt

-rwxrwxr-x 1 amol amol 16912 May 29 14:00 a.out

drwxrwxr-x 3 amol amol 4096 May 28 17:34 kdir

Slip8:

Q.1) Write a C program that redirects standard output to a file output.txt. (use of dup and open system call).

Answer:-

#include <stdio.h>  
#include <stdlib.h>  
#include <unistd.h>  
#include <fcntl.h>  
int main(void) {  
int number1, number2, sum;  
int input\_fds = open("./input.txt", O\_RDONLY);  
if(dup2(input\_fds, STDIN\_FILENO) < 0) {  
printf("Unable to duplicate file descriptor.");  
exit(EXIT\_FAILURE);  
}  
scanf("%d %d", &number1, &number2);  
sum = number1 + number2;  
printf("%d + %d = %d\n", number1, number2, sum);  
return EXIT\_SUCCESS;  
}

Q.2) Implement the following unix/linux command (use fork, pipe and exec system call) ls –l | wc –l.

Answer:-

#include <stdio.h>

#include <stdlib.h>

#include <fcntl.h>

#include<errno.h>

#include<sys/wait.h>

#include <unistd.h>

int main(){

    // array of 2 size a[0] is for

    // reading and a[1] is for

    // writing over a pipe

    int a[2];

    // using pipe for inter

    // process communication

    pipe(a);

    if(!fork())

    {

        // closing normal stdout

        close(1);

        // making stdout same as a[1]

        dup(a[1]);

        // closing reading part of pipe

        // we don't need of it at this time

        close(a[0]);

        // executing ls

        execlp("ls","ls",NULL);

    }

    else

    {

        // closing normal stdin

        close(0);

        // making stdin same as a[0]

        dup(a[0]);

        // closing writing part in parent,

        // we don't need of it at this time

        close(a[1]);

        // executing wc

        execlp("wc","wc",NULL);

    }

}

Output:-

Slip9:-

Q.1) Generate parent process to write unnamed pipe and will read from it

Answer:-

#include <stdio.h>

#include <stdlib.h>

#include <unistd.h>

int main() {

int pipe\_fd[2]; // File descriptors for the pipe

pid\_t child\_pid;

// Create a pipe

if (pipe(pipe\_fd) == -1) {

perror("pipe");

exit(EXIT\_FAILURE);

}

// Create a child process

child\_pid = fork();

if (child\_pid == -1) {

perror("fork");

exit(EXIT\_FAILURE);

}

if (child\_pid == 0) {

// Child process

close(pipe\_fd[1]); // Close the write end in the child process

char buffer[100];

ssize\_t bytes\_read;

// Read from the pipe

bytes\_read = read(pipe\_fd[0], buffer, sizeof(buffer));

if (bytes\_read == -1) {

perror("read");

exit(EXIT\_FAILURE);

}

printf("Child process received: %.\*s", (int)bytes\_read, buffer);

// Close the read end in the child process

close(pipe\_fd[0]);

} else {

// Parent process

close(pipe\_fd[0]); // Close the read end in the parent process

const char \*message = "Hello, child!";

// Write to the pipe

if (write(pipe\_fd[1], message, strlen(message)) == -1) {

perror("write");

exit(EXIT\_FAILURE);

}

// Close the write end in the parent process

close(pipe\_fd[1]);

// Wait for the child process to complete

wait(NULL);

printf("Parent process exiting...\n");

}

return 0;

}

Q.2) 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.

Answer:-

#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;

}

Output:-

$ gcc -o know\_file\_type\_using\_stat know\_file\_type\_using\_stat.c

$ ./know\_file\_type\_using\_stat helloworld.txt

file type is : Regular file

$ ./know\_file\_type\_using\_stat /dev/tty0

file type is : Character device

$ ./know\_file\_type\_using\_stat /dev/sda1

file type is : Block device

$ ./know\_file\_type\_using\_stat $PWD

file type is : Directory

Slip10:-

Q.1) Write a program that illustrates how to execute two commands concurrently with a pipe

Answer:-

#include <stdio.h>

#include <unistd.h>

#include <sys/types.h>

#include <stdlib.h>

int main()

{

int pfds[2];

char buf[30];

if(pipe(pfds)==-1)

{

perror("pipe failed");

exit(1);

}

if(!fork())

{

close(1);

dup(pfds[1];

system (“ls –l”);

}

else

{

printf("parent reading from pipe \n");

while(read(pfds[0],buf,80))

printf("%s \n" ,buf);

}

}

Output:

[student@gcet ~]$ vi pipes2.c

[student@gcet ~]$ cc pipes2.c

[student@gcet ~]$ ./a.out

Parent reading from pipe

Total 24

-rwxrwxr-x l student student 5563Aug 3 10:39 a.out

-rw-rw-r—l

Student student 340 jul 27 10:45 pipe2.c

-rw-rw-r—l student student

Pipes2.c

-rw-rw-r—l student student 401 34127 10:27 pipe2.c

student

Q.2) Generate parent process to write unnamed pipe and will write into it. Also generate child process which will read from pipe

Answer:-

#include <stdio.h>

#include<stdlib.h>

#include

#include<unistd.h>

int main()

{

pid\_t pid;

int r;

char \*ch=NULL;

char \*ch1=NULL;

int readpipe[2];

int writepipe[2];

int a;

int b;

a=pipe(readpipe);

b=pipe(writepipe);

*// check a and b*

pid=fork();

*// check pid*

if(pid==0)

{ *//CHILD PROCESS*

close(readpipe[1]);

close(writepipe[1]);

read(readpipe[0],ch,sizeof(ch));

printf("\nREAD = %s",ch);

close(readpipe[0]);

ch1="YES";

write(writepipe[1],ch1,sizeof(ch1));

close(writepipe[1]);

}

else

{ *//PARENT PROCESS*

close(writepipe[0]);

close(writepipe[1]);

ch="HI!! YOU THERE";

write(readpipe[1],ch,sizeof(ch));

close(readpipe[1]);

read(writepipe[1],ch1,sizeof(ch1));

printf("\nACK RECEIVED %s",ch1);

close(writepipe[1]);

}

return 0;

}

Slip11:-

Slip11:-

Q.1) Write a C program to get and set the resource limits such as files, memory associated with a process

Answer:-

#include <stdio.h>

#include <sys/resource.h>

#include <string.h>

#include <errno.h>

#include <unistd.h>

#include <sys/types.h>

#include <sys/stat.h>

#include <fcntl.h>

int main() {

    struct rlimit old\_lim, lim, new\_lim;

    // Get old limits

    if( getrlimit(RLIMIT\_NOFILE, &old\_lim) == 0)

        printf("Old limits -> soft limit= %ld \t"

           " hard limit= %ld \n", old\_lim.rlim\_cur,

                                 old\_lim.rlim\_max);

    else

        fprintf(stderr, "%s\n", strerror(errno));

    // Set new value

    lim.rlim\_cur = 3;

    lim.rlim\_max = 1024;

    // Set limits

    if(setrlimit(RLIMIT\_NOFILE, &lim) == -1)

        fprintf(stderr, "%s\n", strerror(errno));

    // Get new limits

    if( getrlimit(RLIMIT\_NOFILE, &new\_lim) == 0)

        printf("New limits -> soft limit= %ld "

         "\t hard limit= %ld \n", new\_lim.rlim\_cur,

                                  new\_lim.rlim\_max);

    else

        fprintf(stderr, "%s\n", strerror(errno));

    return 0;

}

Q.2) Write a C program that redirects standard output to a file output.txt. (use of dup and open system call).

Answer:-

#include <stdio.h>  
#include <stdlib.h>  
#include <unistd.h>  
#include <fcntl.h>  
int main(void) {  
int number1, number2, sum;  
int input\_fds = open("./input.txt", O\_RDONLY);  
if(dup2(input\_fds, STDIN\_FILENO) < 0) {  
printf("Unable to duplicate file descriptor.");  
exit(EXIT\_FAILURE);  
}  
scanf("%d %d", &number1, &number2);  
sum = number1 + number2;  
printf("%d + %d = %d\n", number1, number2, sum);  
return EXIT\_SUCCESS;  
}

Slip12:-

Q.1) Write a C program that print the exit status of a terminated child process

Answer:-

#include <stdio.h>

#include <stdlib.h>

#include <unistd.h>

#include <sys/wait.h>

int main() {

pid\_t child\_pid;

// Create a child process

child\_pid = fork();

if (child\_pid == -1) {

perror("fork");

exit(EXIT\_FAILURE);

}

if (child\_pid == 0) {

// Child process

printf("Child process executing...\n");

// Simulate some work in the child process

sleep(2);

// Exit with a specific status

exit(42);

} else {

// Parent process

printf("Waiting for the child process to complete...\n");

// Wait for the child process to complete and get its exit status

int status;

waitpid(child\_pid, &status, 0);

if (WIFEXITED(status)) {

printf("Child process exited with status: %d\n", WEXITSTATUS(status));

} else {

printf("Child process did not exit normally.\n");

}

printf("Parent process exiting...\n");

}

return 0;

}

Q.2) 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, …)

Answer:-

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

#include <sys/stat.h>

// Structure to hold file information

struct FileInfo {

char name[256];

off\_t size;

};

// Function to compare two FileInfo structures based on size

int compareFileSize(const void \*a, const void \*b) {

return ((struct FileInfo \*)a)->size - ((struct FileInfo \*)b)->size;

}

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

if (argc < 2) {

fprintf(stderr, "Usage: %s file1 file2 file3 ...\n", argv[0]);

exit(EXIT\_FAILURE);

}

// Create an array of FileInfo structures

struct FileInfo fileInfos[argc - 1];

// Collect file information

for (int i = 1; i < argc; i++) {

strncpy(fileInfos[i - 1].name, argv[i], sizeof(fileInfos[i - 1].name) - 1);

fileInfos[i - 1].name[sizeof(fileInfos[i - 1].name) - 1] = '\0'; // Ensure null-termination

struct stat fileStat;

if (stat(argv[i], &fileStat) == -1) {

perror("stat");

exit(EXIT\_FAILURE);

}

fileInfos[i - 1].size = fileStat.st\_size;

}

// Sort the array of FileInfo structures based on size

qsort(fileInfos, argc - 1, sizeof(struct FileInfo), compareFileSize);

// Display the sorted filenames and their sizes

printf("File names sorted by size:\n");

for (int i = 0; i < argc - 1; i++) {

printf("%s: %ld bytes\n", fileInfos[i].name, (long)fileInfos[i].size);

}

return 0;

}

Slip13:

Q.1) Write a C program that illustrates suspending and resuming processes using signals

Answer:-

#include <stdio.h>

#include <ospace/unix.h>

int child\_function()

{

while (true) // Loop forever.

{

Printf("Child loop\n");

os\_this\_process::sleep( 1 );

}

return 0; // Will never execute.

}

int main()

{

os\_unix\_toolkit initialize;

os\_process child ( child function ); // Spawn child.

os\_this\_process::sleep( 4 );

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

child.suspend();

printf("Parent sleeps for 4 seconds\n");

os\_this\_process::sleep (4);

printf("child.resume()");

child.resume ();

os\_this\_process::sleep (4);

printf("child.terminate()");

child.terminate ();

printf("Parent finished");

return 0;

}

Output:

Child loop

Child loop

Child loop

Child loop

Child loop

child.suspend()

Parent sleeps for 4 seconds

child.resume()

Child loop

Child loop

Child loop

Child loop

child.terminate()

Child loop

Parent finished

Q.2) 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 foo will return all file names that begins with foo

Answer:-

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

#include <dirent.h>

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

if (argc != 2) {

fprintf(stderr, "Usage: %s <prefix>\n", argv[0]);

exit(EXIT\_FAILURE);

}

// Open the current directory

DIR \*dir = opendir(".");

if (dir == NULL) {

perror("opendir");

exit(EXIT\_FAILURE);

}

// Read directory entries

struct dirent \*entry;

printf("Files starting with '%s':\n", argv[1]);

while ((entry = readdir(dir)) != NULL) {

// Check if the entry is a regular file and starts with the specified prefix

if (entry->d\_type == DT\_REG && strncmp(entry->d\_name, argv[1], strlen(argv[1])) == 0) {

printf("%s\n", entry->d\_name);

}

}

// Close the directory

closedir(dir);

return 0;

}

Slip14:-

Q.1) Display all the files from current directory whose size is greater that n Bytes Where n is accept from user

Answer:-

#include <stdio.h>

#include <stdlib.h>

#include <dirent.h>

#include <sys/stat.h>

void listFilesWithSize(const char \*dirPath, long long fileSizeThreshold) {

DIR \*dir = opendir(dirPath);

if (dir == NULL) {

perror("opendir");

exit(EXIT\_FAILURE);

}

struct dirent \*entry;

struct stat fileStat;

printf("Files with size greater than %lld bytes in the current directory:\n", fileSizeThreshold);

while ((entry = readdir(dir)) != NULL) {

char filePath[1024];

snprintf(filePath, sizeof(filePath), "%s/%s", dirPath, entry->d\_name);

if (stat(filePath, &fileStat) == -1) {

perror("stat");

continue;

}

if (S\_ISREG(fileStat.st\_mode) && fileStat.st\_size > fileSizeThreshold) {

printf("%s (%lld bytes)\n", entry->d\_name, (long long)fileStat.st\_size);

}

}

closedir(dir);

}

int main() {

long long fileSizeThreshold;

printf("Enter the minimum file size in bytes: ");

scanf("%lld", &fileSizeThreshold);

listFilesWithSize(".", fileSizeThreshold);

return 0;

}

Q.2) 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 cal

Answer:-

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

int main(int argc,char\* argv[])  
{  
struct stat info;  
if(argc!=2)  
{  
printf("Enter a filename");  
scanf("%s",argv[1]);  
}  
if(stat(argv[1],&info)==-1)  
{  
printf("stat error/n");  
exit(0);  
}  
printf("inode number=%d\n",info.st\_ino);  
printf("size = %d",(long)info.st\_size);  
printf("last file access = %s\n",ctime(&info.st\_atime));  
printf("notification time = %s\n",ctime(&info.st\_mtime));  
printf("No of Hardlink = %d\n",info.st\_nlink);  
printf("File Permissions : \n");  
printf((info.st\_mode && S\_IRUSR)?"r":"-");  
printf((info.st\_mode && S\_IWUSR)?"w":"-");  
printf((info.st\_mode && S\_IXUSR)?"x":"-");

return 0;

}

Slip15:-

Q.1) Display all the files from current directory whose size is greater that n Bytes Where n is accept from use

Answer:-

#include <stdio.h>

#include <stdlib.h>

#include <dirent.h>

#include <sys/stat.h>

void listFilesWithSize(const char \*dirPath, long long fileSizeThreshold) {

DIR \*dir = opendir(dirPath);

if (dir == NULL) {

perror("opendir");

exit(EXIT\_FAILURE);

}

struct dirent \*entry;

struct stat fileStat;

printf("Files with size greater than %lld bytes in the current directory:\n", fileSizeThreshold);

while ((entry = readdir(dir)) != NULL) {

char filePath[1024];

snprintf(filePath, sizeof(filePath), "%s/%s", dirPath, entry->d\_name);

if (stat(filePath, &fileStat) == -1) {

perror("stat");

continue;

}

if (S\_ISREG(fileStat.st\_mode) && fileStat.st\_size > fileSizeThreshold) {

printf("%s (%lld bytes)\n", entry->d\_name, (long long)fileStat.st\_size);

}

}

closedir(dir);

}

int main() {

long long fileSizeThreshold;

printf("Enter the minimum file size in bytes: ");

scanf("%lld", &fileSizeThreshold);

listFilesWithSize(".", fileSizeThreshold);

return 0;

}

Q.2) 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

Answer:-

#include <signal.h>

#include <stdio.h>

#include <stdlib.h>

#include <sys/types.h>

#include <unistd.h>

// function declaration

void sighup();

void sigint();

void sigquit();

// driver code

void main()

{

    int pid;

    /\* get child process \*/

    if ((pid = fork()) < 0) {

        perror("fork");

        exit(1);

    }

    if (pid == 0) { /\* child \*/

        signal(SIGHUP, sighup);

        signal(SIGINT, sigint);

        signal(SIGQUIT, sigquit);

        for (;;)

            ; /\* loop for ever \*/

    }

    else /\* parent \*/

    { /\* pid hold id of child \*/

        printf("\nPARENT: sending SIGHUP\n\n");

        kill(pid, SIGHUP);

        sleep(3); /\* pause for 3 secs \*/

        printf("\nPARENT: sending SIGINT\n\n");

        kill(pid, SIGINT);

        sleep(3); /\* pause for 3 secs \*/

        printf("\nPARENT: sending SIGQUIT\n\n");

        kill(pid, SIGQUIT);

        sleep(3);

    }

}

// sighup() function definition

void sighup()

{

    signal(SIGHUP, sighup); /\* reset signal \*/

    printf("CHILD: I have received a SIGHUP\n");

}

// sigint() function definition

void sigint()

{

    signal(SIGINT, sigint); /\* reset signal \*/

    printf("CHILD: I have received a SIGINT\n");

}

// sigquit() function definition

void sigquit()

{

    printf("My DADDY has Killed me!!!\n");

    exit(0);

}

Slip16:-

Q.1) Display all the files from current directory which are created in particular month

Answer:-

#include<stdio.h>  
#include<dirent.h>  
#include<string.h>  
#include<sys/stat.h>  
#include<time.h>  
#include<stdlib.h>  
int main(intargc, char \*argv[])  
{  
char in[100],st[100],\*ch,\*ch1,c,buff[512];  
DIR \*dp;  
int i;  
structdirent \*ep;  
struct stat sb;  
charmon[100];  
dp=opendir("./");  
if (dp != NULL)  
{  
while(ep =readdir(dp))  
{  
if(stat(ep->d\_name,&sb) == -1)  
{  
perror("stat");  
exit(EXIT\_SUCCESS);  
}  
strcpy(mon,ctime(&sb.st\_ctime));  
ch=strtok(mon," ");  
ch=strtok(NULL,",");  
ch1=strtok(ch," ");  
if((strcmp(ch1,argv[1]))==0)  
{  
printf("%s\t\t%s",ep->d\_name,ctime(&sb.st\_ctime));  
}  
}  
(void)closedir(dp);  
}  
return 0;  
}

Q.2) 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!!!”

Answer:-

A signal is a type of notification to a process indicating the occurrence of an event. It is also called the software interrupt and is not predictable to know its occurrence. Thus it is also known as an asynchronous event.

A signal can be specified with a name or a number; usually, signal names start with SIG.

To get the signal commands supported by your system, use the “kill -l” (l is for list) command; the following is the list of signal commands.

#include <signal.h>

#include <stdlib.h>

#include <unistd.h>

#include <sys/types.h>

// function declaration of sighup, sigint and sigquit functions

void sighup();

void sigint();

void sigquit();

// main function or driver code

void main()

{

int pid;

// pid variable, which will be used later to identify the process, whether it is child process or parent process

// to get the child process

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

{

perror("fork");

exit(1);

}

if (pid == 0)

{  /\* child process, since pid equals to zero for child process \*/

signal(SIGHUP, sighup);

signal(SIGINT, sigint);

signal(SIGQUIT, sigquit);

for (;;)

; /\* infinite loop i.e. loop for ever  \*/

}

else /\* parent process\*/

{ // pid hold the process id of child process

printf("\nPARENT: sending SIGHUP\n\n");

kill(pid, SIGHUP);

sleep(3); // pause for 3 seconds

printf("\nPARENT: sending SIGINT\n\n");

kill(pid, SIGINT);

sleep(3); // pause for 3 seconds

printf("\nPARENT: sending SIGQUIT\n\n");

kill(pid, SIGQUIT);

sleep(3); // pause for 3 seconds

}

}

// function definition of sighup()

void sighup()

{

signal(SIGHUP, sighup); /\* reset signal \*/

printf("CHILD: I have received a SIGHUP\n");

}

// function definition of sigint()

void sigint()

{

signal(SIGINT, sigint); /\* reset signal \*/

printf("CHILD: I have received a SIGINT\n");

}

// function definition of sigquit()

void sigquit()

{

printf("My DADDY has Killed me!!!\n");

exit(0);

}