**UNIX SOLUTIONS**

**12. b. Write and execute Unix system program to create co-process which performs Concatenation of two strings.**

#include<sys/stat.h>

#include<sys/types.h>

#include<fcntl.h>

#include<unistd.h>

#include<stdio.h>

#include<stdlib.h>

#include<string.h>

#define maxline 2000

int main()

{

int n;

char int1[maxline],int2[maxline];

char line[maxline];

char \*str;

while((n=read(STDIN\_FILENO, line,maxline))>0)

{

line[n]=0;

if(sscanf(line,"%s%s",int1,int2)==2)

{

str=strcat(int1,int2);

sprintf(line,"%s\n",str);

n=strlen(line);

if(write(STDOUT\_FILENO,line,n)!=n)

printf("write error");

}

else

{

if(write(STDOUT\_FILENO,"invalid arg\n",13)!=13)

printf("write error");

}}

exit(0);

}

**12 a. Write a program to register the exit handlers and demonstrate their order of invocation**

#include <stdio.h>

#include <unistd.h>

void hander\_1()

{

printf("Exit Hander - I \n");

}

void hander\_2()

{

printf("Exit Hander - II \n");

}

int main()

{

atexit(hander\_1);

atexit(hander\_2);

printf("Calling Implicit exit() \n");

return 0;

}

**11 b. Write a program to implement reader-writer problem using semaphore. The reader process reads a value that is written by writer process and both of them execute in mutual exclusion.**

**6b. Write a program to implement reader-writer problem using semaphore. The reader process reads a value that is written by writer process and both of them execute in mutual exclusion**

#include<stdio.h>

#include<semaphore.h>

#include<pthread.h>

#include<stdlib.h>

#include<time.h>

pthread\_t tidP[20],tidC;

sem\_t full,empty,mutex;

int counter,m,k,buffersize;

int buffer[100];

void initialize()

{

sem\_init(&full,0,0);

sem\_init(&empty,0,buffersize);

sem\_init(&mutex,0,1);

counter=0;

}

void write(int item)

{

buffer[counter++]=item;

}

int read()

{

return(buffer[--counter]);

}

void \* producer (void \* param)

{

srand(time(NULL));

for(k=0;k<=10;k++)

{

m=rand()%100;

sem\_wait(&empty);

sem\_wait(&mutex);

write(m);

printf("\nWriter wrote %d",m);

sem\_post(&mutex);

sem\_post(&full);

}

}

void \* consumer (void \* param)

{

int i;

for(k=0;k<=10;k++)

{

sem\_wait(&full);

sem\_wait(&mutex);

i=read();

printf("\nReader read %d",i);

sem\_post(&mutex);

sem\_post(&empty);

}

}

int main()

{

buffersize=1;

int n1,n2,i;

initialize();

n1=1;

for(i=0;i<n1;i++)

pthread\_create(&tidP[i],NULL,producer,NULL);

pthread\_create(&tidC,NULL,consumer,NULL);

for(i=0;i<n1;i++)

pthread\_join(tidP[i],NULL);

pthread\_join(tidC,NULL);

exit(0);

}

**11 a. Write a program such that it initializes itself as a daemon Process.**

#include<stdio.h>

#include<stdlib.h>

#include<sys/stat.h>

#include<sys/types.h>

#include<unistd.h>

int deamon\_init()

{

pid\_t pid;

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

return -1;

else if(pid!=0)

exit(0);

setsid();

chdir("\\");

umask(0);

return (0);

}

int main()

{

int x=deamon\_init();

system("ps -axj");

}

**10 b. Write a program (use signal system call)**

1. **which calls a signal handler on SIGINT signal and then reset the default action of the SIGINT signal**
2. **which ignores SIGINT signal and then reset the default action of SIGINT signal**

**i)**

#include <signal.h>

#include <stdio.h>

#include <unistd.h>

void gotit(int sig)

{

printf("Got the signal!!!\n");

(void)signal(SIGINT,SIG\_DFL);

}

int main()

{

(void)signal(SIGINT,gotit);

while(1)

{

printf("Hello World\n");

sleep(1);

}

}

**ii)**

#include <signal.h>

#include <stdio.h>

#include <unistd.h>

int main()

{

(void)signal(SIGINT,SIG\_IGN);

int count=0;

while(1)

{

printf("Hello World\n");

count++;

if(count>10)

(void)signal(SIGINT,SIG\_DFL);

sleep(1);

}

}

**10a.Write a program in which the parent displays uppercase characters A-Z and child displays lowercase characters a-z with a delay of max of 2 seconds between each character. Run the program repeatedly and see if there is racing.**

#include <stdio.h>

#include <unistd.h>

int main()

{

pid\_t pid;

int i;

pid = fork();

if(pid)

{

for(i=0;i<26;i++)

{

printf("Parent : %c\n", (char)(65+i));

sleep(2);

}

}

if(!pid)

{

for(i=0;i<26;i++)

{

printf("Child : %c\n", (char)(97+i));

sleep(2);

}

}

}

**9b. Write a program that executes an interpreter file, which calls another program to display the arguments that is passed in the exec function**

**inter.c**

#include<sys/stat.h>

#include<sys/types.h>

#include<fcntl.h>

#include<unistd.h>

#include<stdio.h>

int main()

{

pid\_t pid;

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

printf("error");

else if(pid==0)

if(execl("textinterpreter","test","myarg1","myarg2","myarg4",(char \*)0)<0)

printf("error1");

if(waitpid(pid,NULL,0)<0)

printf("wait eror");

system("ls> list");

return 0;

}

**textinterpreter**

#! /home/guest1/echoarg my2

**echoarg.c**

#include<sys/stat.h>

#include<sys/types.h>

#include<fcntl.h>

#include<unistd.h>

#include<stdio.h>

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

{

int i;

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

{

printf("argv[%d]=%s",i,argv[i]);

}

}

**output**

cc echoarg.c -o echoarg

chmod 777 textinterpreter

gcc inter.c

./a.out

**9a. Write a program to:**

* 1. **Create a zombie process**
  2. **Create an orphan process**

#include <stdio.h>

#include <unistd.h>

#include<stdlib.h>

void zombie()

{

pid\_t pid = fork();

if(!pid) // child

{

exit(0);

}

else // parent

{

system("ps -aux | grep Z"); // search for zombies

}

}

void orphan()

{

pid\_t pid = fork();

if(!pid)

{

sleep(10);

printf("Adoptive Parent (init --user): %d\n", getppid());

}

else

{

printf("Original Parent : %d\n", getpid());

exit(0);

}

}

int main()

{

int ch;

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

scanf("%d",&ch);

if(ch == 1)

zombie();

else

orphan();

return 0;

}

**8b. Write a program to setup a timer and setup handler for the timer using fork (or exec) to check who gets the timer signal**

#include <stdio.h>

#include <unistd.h>

#include <signal.h>

void handler()

{

printf("PID calling handler : %d\n",getpid());

}

int main()

{

pid\_t pid = fork();

signal(SIGALRM,handler);

alarm(5);

if(pid)

{

printf("PID of Parent : %d\n",getpid());

sleep(10);

}

else if(!pid)

{

printf("PID of child : %d\n", getpid());

sleep(10);

}

}

**8a. Write a program to illustrate effect of setjmp and longjmp functions on register and volatile variables.**

#include <stdio.h>

#include <setjmp.h>

jmp\_buf jmp;

int divide(int a,int b)

{

if (b == 0)

{

longjmp(jmp,1);

}

else

return a/b;

}

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

{

int ans;

int a = atoi(argv[1]);

int b = atoi(argv[2]);

if(setjmp(jmp) == 0)

{

ans = divide(a,b);

printf("Quotient : %d\n",ans);

}

else

{

printf("Divide by Zero Exception Caught!\n");

}

return 0;

}

**7b. Write a program to demonstrate inter process communication using Shared memory. One process should enter all the alphabets into the shared memory and other process should read it**

**//SERVER**

#include <sys/types.h>

#include <sys/ipc.h>

#include <sys/shm.h>

#include <stdio.h>

#include <stdlib.h>

#define MAXSIZE 27

void die(char \*s)

{

perror(s);

exit(1);

}

int main()

{

char c;

int shmid;

key\_t key;

char \*shm, \*s;

key = 5678;

if ((shmid = shmget(key, MAXSIZE, IPC\_CREAT | 0666)) < 0)

die("shmget");

if ((shm = shmat(shmid, NULL, 0)) == (char \*) -1)

die("shmat");

/\* Put some things into the memory for the other process to read. \*/

s = shm;

for (c = 'a'; c <= 'z'; c++)

\*s++ = c;

/\* Wait until the other process changes the first character of our memory to '\*', indicating that it has read what we put there. \*/

while (\*shm != '\*')

sleep(1);

exit(0);

}

**//CLIENT**

#include <sys/types.h>

#include <sys/ipc.h>

#include <sys/shm.h>

#include <stdio.h>

#include <stdlib.h>

#define MAXSIZE 27

void die(char \*s)

{

perror(s);

exit(1);

}

int main()

{

int shmid;

key\_t key;

char \*shm, \*s;

key = 5678;

if ((shmid = shmget(key, MAXSIZE, 0666)) < 0)

die("shmget");

if ((shm = shmat(shmid, NULL, 0)) == (char \*) -1)

die("shmat");

//Now read what the server put in the memory.

for (s = shm; \*s != '\0'; s++)

putchar(\*s);

putchar('\n');

/\* Change the first character of the segment to '\*', indicating we have read the segment. \*/

\*shm = '\*';

exit(0);

}

**7a. Write a program to check whether user is valid or not**

#include <stdio.h>

#include <unistd.h>

#include <stdlib.h>

#include <sys/types.h>

#include <pwd.h>

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

{

struct passwd \*pw;

if(pw = getpwnam(argv[1]))

{

printf("Login Name : %s\n",pw->pw\_name);

printf("Shell : %s\n", pw->pw\_shell);

printf("UID : %d\n", pw->pw\_uid);

printf("GID : %d\n", pw->pw\_gid);

}

else

{

printf("User %s does not exist\n",argv[1]);

}

}

**6a. Write a program to Copy access and modification time of a file to another file using utime function**

#include <stdio.h>

#include <sys/stat.h>

#include <sys/types.h>

#include <unistd.h>

#include <utime.h>

#include <time.h>

#include <fcntl.h>

int main(int argc,char\* argv[]) //copying ctime and mtime of argv[2] to argv[1]

{

int fd;

struct stat statbuf\_1;

struct stat statbuf\_2;

struct utimbuf times;

if(stat(argv[1],&statbuf\_1)<0)

printf("Error!\n");

if(stat(argv[2],&statbuf\_2)<0)

printf("Error!\n");

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

printf("Access Time %s\nModification Time%s\n",ctime(&statbuf\_1.st\_atime),ctime(&statbuf\_1.st\_mtime));

times.modtime = statbuf\_2.st\_mtime;

times.actime = statbuf\_2.st\_mtime;

if(utime(argv[1],&times)<0)

printf("Error copying time \n");

if(stat(argv[1],&statbuf\_1)<0)

printf("Error!\n");

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

printf("Access Time %s\nModification Time%s\n",ctime(&statbuf\_1.st\_atime),ctime(&statbuf\_1.st\_mtime));

}

**5b. Write a program using *sigaction* system call which calls a signal handler on *SIGINT* signal and then reset the default action of the *SIGINT* signal**

#include <signal.h>

#include <stdio.h>

#include <unistd.h>

struct sigaction new\_sig;

void gotit(int sig)

{

printf("Got the signal!!!\n");

new\_sig.sa\_handler = SIG\_DFL;

sigaction(SIGINT,&new\_sig,0);

}

int main()

{

new\_sig.sa\_flags=0;

sigemptyset(&new\_sig.sa\_mask);

sigaddset(&new\_sig.sa\_mask, SIGINT);

new\_sig.sa\_handler = gotit;

sigaction(SIGINT,&new\_sig,0);

while(1)

{

printf("Hello World\n");

sleep(1);

}

}

**5a.Write a program to Remove an empty files from the given directory**

#include <stdio.h>

#include <fcntl.h>

#include <unistd.h>

#include <dirent.h>

int main()

{

DIR \*dp;

struct dirent \*dir;

int fd,n;

dp = opendir("."); //open current directory

if(dp)

{

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

{

fd = open(dir->d\_name,O\_RDWR,0777);

n = lseek(fd,0,SEEK\_END);

if(!n)

{

unlink(dir->d\_name);

}

}

}

}

**4b. Write a program to implement ls –li command which list the files in a specified directory. Your program should Print 5 attributes of files.**

#include <stdio.h>

#include <unistd.h>

#include <fcntl.h>

#include <dirent.h>

#include <time.h>

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

{

struct dirent \*dir;

struct stat mystat;

DIR \*dp;

dp = opendir(".");

if(dp)

{

while(dir = readdir(dp))

{

stat(dir->d\_name,&mystat);

// inode mode uid guid access\_time

printf("%ld %o %d %d %s %s\n",

mystat.st\_ino,mystat.st\_mode,mystat.st\_uid,mystat.st\_gid,ctime(&mystat.st\_atime),dir->d\_name);

}

}

}

**4a. Write a program to create a co-process which performs addition of two numbers.**

#include<sys/stat.h>

#include<sys/types.h>

#include<fcntl.h>

#include<unistd.h>

#include<stdio.h>

#include<stdlib.h>

#include<string.h>

#define maxline 2000

int main()

{

int n,int1,int2;

char line[maxline];

while((n=read(STDIN\_FILENO, line,maxline))>0)

{

line[n]=0;

if(sscanf(line,"%d%d",&int1,&int2)==2)

{

sprintf(line,"%d\n",int1+int2);

n=strlen(line);

if(write(STDOUT\_FILENO,line,n)!=n)

printf("write error");

}

else

{

if(write(STDOUT\_FILENO,"invalid arg\n",13)!=13)

printf("write error");

}}

exit(0);

}

**3b. Write a program to demonstrate inter process communication using Message Queue. One process should send the name of the file and the other process should return its contents**

**//SERVER**

#include<stdio.h>

#include<unistd.h>

#include<fcntl.h>

#include<sys/types.h>

#include<sys/ipc.h>

#include<sys/msg.h>

#include<sys/stat.h>

#include<string.h>

#define KEY 600

struct msg

{

long int type;

char a[1024];

int pid;

}p;

int main()

{

int m,n,fd,m1;

m=msgget(KEY,0666|IPC\_CREAT);

while(1)

{

msgrcv(m,&p,sizeof(p),1,0);

printf("Filename from client %s\n",p.a);

fd=open(p.a,O\_RDONLY);

n=read(fd,p.a,1024);

p.type=p.pid;

p.pid=getpid();

msgsnd(m,&p,sizeof(p),0);

}

}

**//Client**

#include<stdio.h>

#include<unistd.h>

#include<fcntl.h>

#include<sys/types.h>

#include<sys/stat.h>

#include<string.h>

#define KEY 600

struct msg

{

long int type;

char a[1024];

int pid;

}p,p1;

int main()

{

int m;

m=msgget(KEY,0);

p.type=1;

printf("\nEnter the file name: ");

scanf("%s",&p.a);

pid\_t pid;

p.pid=getpid();

msgsnd(m,&p,sizeof(p),0);

msgrcv(m,&p1,sizeof(p),p.pid,0);

printf("%s",p1.a);

}

**3a. Write a program to display various details of a file using stat structure(At least 5 fields) .**

#include <stdio.h>

#include <unistd.h>

#include <sys/stat.h>

#include <sys/types.h>

#include <fcntl.h>

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

{

struct stat mystat;

stat(argv[1],&mystat);

printf("Device ID %ld\n",mystat.st\_dev);

printf("Inode Number %ld\n",mystat.st\_ino);

printf("User ID %d\n",mystat.st\_uid);

printf("Group ID %d\n",mystat.st\_gid);

printf("File Size %ld\n",mystat.st\_size);

printf("Block Size %ld\n",mystat.st\_blksize);

printf("Protection : %o\n",mystat.st\_mode);

}

**2b. Write a program to implement client - server communication using pipes. Client should send a string to the server and it should change the first character of the string to uppercase and sends it back to client**

**//client**

#include <stdio.h>

#include <unistd.h>

#include <sys/stat.h>

#include <sys/types.h>

#include <string.h>

#include <fcntl.h>

#define MAX 80

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

{

char buffer[MAX];

int in,out;

int n;

in = open("server\_to\_client",O\_RDWR,0777);

out = open("client\_to\_server",O\_RDWR,0777);

printf("Sending Message\n");

n = strlen(argv[1]);

write(out,argv[1],n);

read(in,buffer,MAX);

printf("Message Received : %s\n",buffer);

close(in);

close(out);

}

**//server**

#include <stdio.h>

#include <unistd.h>

#include <sys/stat.h>

#include <sys/types.h>

#include <fcntl.h>

#include <string.h>

#define MAX 80

int main()

{

char buffer[MAX];

int in,out,n;

mkfifo("server\_to\_client",0777);

mkfifo("client\_to\_server",0777);

while(1)

{

in = open("client\_to\_server",O\_RDWR,0777);

out = open("server\_to\_client",O\_RDWR,0777);

memset(buffer,0,MAX);

printf("Waiting for Message\n");

n = read(in,buffer,MAX);

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

buffer[0] = toupper(buffer[0]);

printf("Sending Reply\n");

write(out,buffer,n);

close(in);

close(out);

}

return 0;

}

**2a. Write a program to display the file content in reverse order using lseek system call.**

#include <stdio.h>

#include <unistd.h>

#include <fcntl.h>

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

{

int fd , n;

int i; // loop variable

char buffer[1];

fd = open(argv[1],O\_CREAT|O\_RDWR,0777);

n = lseek(fd,0,SEEK\_END); //size of the input file

for(i=1;i<=n;i++)

{

lseek(fd,-i,SEEK\_END);

read(fd,buffer,1);

write(STDOUT\_FILENO,buffer,1);

}

printf("\n");

return 0;

}

**1b. Write a program to create client and server communication, where message sent by the client should be received by the server (Using FIFO System call).**

**//SERVER**

#include<stdio.h>

#include<unistd.h>

#include<fcntl.h>

#include<sys/types.h>

#include<sys/stat.h>

#include<string.h>

int main()

{

char msg[25]="";

mkfifo("fifo1",0600);

int fd;

fd=open("fifo1",O\_RDONLY);

printf("\nMessage sent from client is \n");

while(read(fd,msg,25))

{

printf("%s",msg);

}

}

**//CLIENT**

#include<stdio.h>

#include<unistd.h>

#include<fcntl.h>

#include<sys/types.h>

#include<sys/stat.h>

#include<string.h>

int main()

{

char msg[100]="";

int fd;

fd=open("fifo1",O\_WRONLY);

printf("\nEnter the message to be sent to server:");

scanf("%s",msg);

write(fd,msg,strlen(msg));

}

**1a.Write a program to read *n* characters and store them into the specified file and redirect the output to file using dup2 function**

#include <stdio.h>

#include <unistd.h>

#include <fcntl.h>

#define SIZE 4096

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

{

char buffer[SIZE];

int n = 0;

int fd1 , fd2 ;

fd1 = open(argv[1],O\_CREAT|O\_RDWR,0777);

fd2 = open(argv[2],O\_CREAT|O\_RDWR,0777);

dup2(fd2,fd1);

n = read(STDIN\_FILENO,buffer,SIZE);

if(write(fd1,buffer,n)!=n)

printf("Error Writing!\n");

return 0;

}