Assign. No. 20. Write a C program to create ‘n’ child processes. When all ‘n’ child processes terminate, Display total cumulative time children spent in user and kernel mode.

```
#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("\n fork 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);
} }
```

Assign No. 21. 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”

```
#include<stdio.h>
#include<unistd.h>
int main()
{
int pipefds[2];
int returnstatus;
char writemessages[3][20]={"Hello World", "Hello SPPU","Linux isFunny"};
char readmessage[20];
returnstatus = pipe(pipefds);
if (returnstatus == -1) {
printf("Unable to create pipe\n");
return 1;
}
int child = fork();
if(child==0){
printf("Child is Writing to pipe - Message 1 is %s\n", writemessages[0]);
write(pipefds[1], writemessages[0], sizeof(writemessages[0]));
printf("Child is Writing to pipe - Message 2 is %s\n", writemessages[1]);
write (pipefds[1],writemessages[1], sizeof(writemessages[1]));
printf("Child is Writing to pipe - Message 3 is %s\n", writemessages[2]);
write(pipefds[1], writemessages[2], sizeof(writemessages[2]));
}
Else {
read(pipefds[0], readmessage, sizeof(readmessage));
printf("Parent Process is Reading from pipe – Message 1 is %s\n",readmessage);
read(pipefds[0], readmessage, sizeof(readmessage));
printf("Parent Process is Reading from pipe – Message 2 is %s\n",readmessage);
read(pipefds[0], readmessage, sizeof(readmessage));
printf("Parent Process is Reading from pipe – Message 3 is %s\n",readmessage);
}
```

Assign No. 26 Write a C program which creates a child process and child process catches asignal 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 Papahas Killed me!!!".

```
#include<stdio.h>
#include<signal.h>
#include<stdlib.h>
void sighup();
void sigint();
void sigquit();
main()
{
int pid,i,j,k;
if ((pid = fork() ) < 0)
{
perror("fork");
exit(1);
}
if ( pid == 0)
{
signal(SIGHUP,sighup);
signal(SIGINT,sigint);
signal(SIGQUIT,sigquit);
for(;;);
}
else
{ j=0;
for(i=1;i<=5;i++)
{ j++;
printf("PARENT: sending SIGHUP Signal : %d\n",j);
```

```
kill(pid,SIGHUP);
sleep(3);
printf(:PARENT : Sending Signal :%d\n”,j);
kill (pid,SIGINT);
sleep(3);
}
sleep(3);
printf("Parent sending SIGQUIT\n");
kill(pid,SIGQUIT);
}
}
void sighup()
{
signal(SIGHUP,sighup);
printf("Child: I have received sighup\n");
}
void sigint()
{
signal(SIGINT,sigint);
printf("Child: I have received sighINT\n");
}
void sigquit()
{
printf("My daddy has killed me\n");
exit(0);
}
```

Assign No. 27. Write a C program to send SIGALRM signal by child process to parentprocess and parent process make a provision to catch the signal and display alarm is fired.(Use Kill, fork, signal and sleep system call)

```
#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[])

{  if (argc!=3)

{

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

}

int PauseSecond=(argv[1]);

{

if(fork()==0)

{

printf("waiting for alarm to go off\n");

printf("%dsecond pause",PauseSecond);

sleep(PauseSecond);

 kill(getpid(),SIGALRM);

}

else{

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

```
signal(SIGALRM,Dingdong);

printf("done");

}}}
```

Assign No.19 Implement the following unix/linux command (use fork, pipe and exec system call) ls –l | wc –l

//ls-l program

```
#include <stdio.h>
#include <unistd.h>
#include <sys/types.h>
#include <sys/wait.h>
#include <stdlib.h>
int main()
{
int pid;            //process id
pid = fork();     //create another process
if ( pid < 0 )
{                              //fail
printf("\nFork failed\n");
exit (-1);
}
else if ( pid == 0 )
{                   //child
execlp ( "/bin/ls", "ls", "-l", NULL );  //execute ls
}
else
{                          //parent
wait (NULL);              //wait for child
printf("\nchild complete\n");
exit (0);
}
}
```

Assign No. 25 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.

```
#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);
}
```