**DAY -23 (28-11-2024)**

WAIT:

* Don’t write wait in the child process
* Program:

#include <stdio.h>

#include <stdlib.h>

#include <sys/types.h>

#include <sys/wait.h>

#include <unistd.h>

int main()

{

printf("\nI am in main\n");

pid\_t id = fork();

if(id == 0)

{

printf("\nI am Child PID: %d\n",getpid());

printf("\nChild's Parent id PPID: %d",getppid());

printf("\nChild's Process id = %d",id);

}

else

{

wait(0);

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

printf("\nParent's PPID: %d",getppid());

printf("\nChild's Process id = %d",id);

}

printf("\nHello World %d\n",getppid());

printf("\n\n");

return 0;

}

* EXEC:
* Execute command line
* Ps ,ls command
* extern char \*\*environ;
* int execl(const char \*path, const char \*arg, ...

/\* (char \*) NULL \*/);

* int execv(const char \*path, char \*const argv[]);
* The exec() family of functions replaces the current process image with a new process image.
* The execv(), execvp(), and execvpe() functions provide an array of pointers to null-terminated strings that represent the argument list available to the new program.
* The exec() functions return only if an error has occurred. The return value is -1, and errno is set to indicate the error.
* Program 1:

#include<stdio.h>

2 #include<stdlib.h>

3 #include<sys/types.h>

4 #include<sys/wait.h>

#include<unistd.h>

int main()

{

printf("\n This is sprog4");

if(fork()>0)

{

wait(0);

printf("\n sprog4 PID : %d \n", getpid());

}

else

{

execl("/bin/ls","/bin/ls",(char\*)0);

}

printf("ending sprog4");

printf("\n\n");

return 0;

}

Ouput:

1.c 3.c a.out

This is sprog4

sprog PID : 11308

ending sprog4

program 2:

#include<stdio.h>

2 #include<stdlib.h>

3 #include<sys/types.h>

4 #include<sys/wait.h>

5 #include<unistd.h>

6 int main()

7 {

8 printf("\n This is sprog4");

9

10 execl("/bin/ls","/bin/ls",(char\*)0); // path ,which command,argument

11

12 printf("ending sprog4");

13 printf("\n\n");

14 return 0;

15 }

Output:

1.c 3.c a.out

Program 3:

In line 10 we can write execl(“/bin/ls”,”ls”,(char\*)0);

To the same output

But we have to give the path clearly

Program 4:

#include<stdio.h>

2 #include<stdlib.h>

3 #include<sys/types.h>

4 #include<sys/wait.h>

5 #include<unistd.h>

6 int main()

7 {

8 printf("\n This is sprog4");

9

10 if( execl("ls","ls",(char\*)0)<0)

11 {

12 perror("execl:");

13 exit(EXIT\_FAILURE);

14 }

15

16

17 printf("ending sprog4");

18 printf("\n\n");

19 return 0;

20 }

Output:

execl:: No such file or directory

Program 5:

#include<stdio.h>

2 #include<stdlib.h>

3 #include<sys/types.h>

4 #include<sys/wait.h>

5 #include<unistd.h>

6 int main()

7 {

8 printf("\n This is sprog4");

9

10 if( execl("/bin/ls","ls",(char\*)0)<0)

11 {

12 perror("execl:");

13 exit(EXIT\_FAILURE);

14 }

15

16

17 printf("ending sprog4");

18 printf("\n\n");

19 return 0;

20 }

Output: 1.c 3.c a.out

#include<stdio.h>

#include<stdlib.h>

#include<sys/types.h>

#include<sys/wait.h>

#include<unistd.h>

Write a program calculator using command line arguments and supply operation and operands

Vi calc.c 🡺 switch case

Gcc -o calc calc.c 🡺 calc.o

Vi ex.c

Gcc-o ex1 ex.c 🡺ex1

./ex1 /home2/user44/day23/calc /home22/user44/day23/calc 2 3 +

Vi calc.c

Program:

#include<stdio.h>

3 #include<stdlib.h>

4 #include<sys/types.h>

5 #include<sys/wait.h>

6 #include<unistd.h>

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

8 {

9 int c;

10 char oper;

11 int a=atoi(argv[1]);

12 int b=atoi(argv[2]);

13 oper=argv[3][0];

14 switch(oper)

15 {

16 case '+':

17 c=a+b;

18 printf("%d\n",c);

19 break;

20 case '-':

21 c=a-b;

22 printf("%d\n",c);

23 break;

24 case '\*':

25 c=a\*b;

26 printf("%d\n",c);

27 break;

28 case '/':

29 c=a/b;

30 printf("%d\n",c);

31 break;

32 }

33

34 }

Vi ex.c

#include<stdio.h>

2 #include<stdlib.h>

3 #include<sys/types.h>

4 #include<sys/wait.h>

5 #include<unistd.h>

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

7 {

8 execl(argv[1],argv[2],argv[3],argv[4],argv[5],(char\*)0);

9 printf("\n\n");

10 }

* EXECV:
* Program: vi exv.c

#include <stdio.h>

#include <stdlib.h>

#include <sys/types.h>

#include <unistd.h>

#include <string.h>

#include <sys/wait.h>

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

{

char \*temp[5];

for(int i=0;i<5;i++)

temp[i] = (char \*)malloc(1024);

// temp[0] = "ls";

// temp[1] = "-l";

// temp[2] = (char \*)0;

// execv("/usr/bin/ls", temp);

if(strcmp(argv[1],"calc")==0)

{

strcpy(temp[0], argv[1]);

strcpy(temp[1], argv[2]);

strcpy(temp[2], argv[3]);

strcpy(temp[3], argv[4]);

temp[4] = (char \*)0;

}

else if(strcmp(argv[1],"area")==0)

{

//area

strcpy(temp[0], argv[1]);

strcpy(temp[1], argv[2]);

strcpy(temp[2], argv[3]);

// strcpy(temp[3], argv[4]);

temp[4] = (char \*)0;

}

execv(argv[1], temp);

printf("\nThis will not print at all\n\n");

return 0;

}

Gcc -o ec exv.c

./ec /home2/user44/day23/calc calc 1 2 +

* ENVP:
* Program :

#include<stdio.h>

2 #include<stdlib.h>

3 #include<sys/types.h>

4 #include<sys/wait.h>

5 #include<unistd.h>

6

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

8 {

9 for(int i=0;envp[i];i++)

10 {

11 printf("%s\n",envp[i]);

12 }

13 return 0;

14 }

15

Program 2:

#include<stdio.h>

#include<stdlib.h>

#include<sys/types.h>

#include<sys/wait.h>

#include<unistd.h>

#include<string.h>

7 extern char\*\*environ;

8 int main()

9 {

10 for(int i=0;environ[i];i++)

11 {

12 printf("%s\n",environ[i]);

13 }

14 return 0;

15 }

* WAIT:
* WNOHANG return immediately if no child has exited.
* SIGNAL:
* Program:

/\*

Program to show on waitpid()

\*/

#include <stdio.h>

#include <stdlib.h>

#include <sys/types.h>

#include <unistd.h>

#include <sys/wait.h>

#include <string.h>

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

{

pid\_t cpid1, cpid2,w;

int status;

cpid1 = fork();

if(cpid1 == -1)

{

printf("\nUnable to create the child process by cpid1\n");

exit(EXIT\_FAILURE);

}

else if(cpid1 == 0)

{

printf("\nFirst child pid: %d", getpid());

while(1);

// if(argc == 1)

// pause();

// \_exit(atoi(argv[1]));

}

else

{

// wait(0);

do{

w = waitpid(cpid1, &status,WUNTRACED|WCONTINUED);

if(w == -1)

{

perror("waitpid() ");

\_exit(EXIT\_FAILURE);

}

if(WIFEXITED(status))

{

printf("\nStatus : %d", WEXITSTATUS(status));

}

else if(WIFSIGNALED(status))

{

printf("\nKilled by the signal: %d", WTERMSIG(status));

}

else if(WIFSTOPPED(status))

{

printf("\nStoped by Signal %d",WSTOPSIG(status));

}

else if(WIFCONTINUED(status))

{

printf("\nContinued");

}

else

{

printf("\nUnCaught signal");

}

}while(!WIFSIGNALED(status) && !WIFEXITED(status));

printf("\nEnd of Parent\n");

// exit(EXIT\_SUCCESS);

}

printf("\n\n");

return 0;

}

* #include <signal.h>

typedef void (\*sighandler\_t)(int);

sighandler\_t signal(int signum, sighandler\_t handler);

* The signals SIGKILL and SIGSTOP cannot be caught or ignored.
* RETURN VALUE

signal() returns the previous value of the signal handler, or SIG\_ERR on error. In the event of an error, errno is set to indicate the cause.

* Question: write a menu based program to calculate area of the circle, rectangle , etc while you have displayed the menu if the signal is interrupted you have handled it

If you enter the menu for example rectangle while entering the if suppose you enter ctrl +c first clean up and then again the menu have to display

If you enter zero in switch it have to terminate in that cleaning of the memory and come out of the program