**Day-23**

* Child must die first, then parent need to die.

**KILL:**

**kill - send a signal to a process**

**user69@trainux01:~/day21$ kill -l**

1) SIGHUP 2) SIGINT 3) SIGQUIT 4) SIGILL 5) SIGTRAP

6) SIGABRT 7) SIGBUS 8) SIGFPE 9) SIGKILL 10) SIGUSR1

11) SIGSEGV 12) SIGUSR2 13) SIGPIPE 14) SIGALRM 15) SIGTERM

16) SIGSTKFLT 17) SIGCHLD 18) SIGCONT 19) SIGSTOP 20) SIGTSTP

21) SIGTTIN 22) SIGTTOU 23) SIGURG 24) SIGXCPU 25) SIGXFSZ

26) SIGVTALRM 27) SIGPROF 28) SIGWINCH 29) SIGIO 30) SIGPWR

31) SIGSYS 34) SIGRTMIN 35) SIGRTMIN+1 36) SIGRTMIN+2 37) SIGRTMIN+3

38) SIGRTMIN+4 39) SIGRTMIN+5 40) SIGRTMIN+6 41) SIGRTMIN+7 42) SIGRTMIN+8

43) SIGRTMIN+9 44) SIGRTMIN+10 45) SIGRTMIN+11 46) SIGRTMIN+12 47) SIGRTMIN+13

48) SIGRTMIN+14 49) SIGRTMIN+15 50) SIGRTMAX-14 51) SIGRTMAX-13 52) SIGRTMAX-12

53) SIGRTMAX-11 54) SIGRTMAX-10 55) SIGRTMAX-9 56) SIGRTMAX-8 57) SIGRTMAX-7

58) SIGRTMAX-6 59) SIGRTMAX-5 60) SIGRTMAX-4 61) SIGRTMAX-3 62) SIGRTMAX-2

63) SIGRTMAX-1 64) SIGRTMAX

**EXEC :**

execl, execlp, execle, execv, execvp, execvpe - execute a file

* Special semantics for execlp() and execvp().
* user69@trainux01:~$ date

Thu Nov 28 06:40:10 UTC 2024

* To print list of files

|  |
| --- |
| #include <stdio.h>  #include<stdlib.h>  #include<sys/types.h>  #include<unistd.h>  #include<sys/wait.h>  int main() {  printf("\nThis is a program\n");  if(fork() >0)  {  wait(0);  printf("\nsPrg04 PID: %d",getpid());  }  else  {  execl("/bin/ls","/bin/ls",(char \*)0);  }  printf("\nEnding sprog04");  printf("\n\n");  return 0;  }  Output :  This is a program  a.out t1.c t2.c t3.c  sPrg04 PID: 24136  Ending sprog04 |

* ./ec2

|  |
| --- |
| #include <stdio.h>  #include<stdlib.h>  #include<sys/types.h>  #include<unistd.h>  #include<sys/wait.h>  int main(int argc, char \*argv[])  {  printf("\nec2.c\n");  printf("\nPID: %d",getpid());  printf("\n%s\n%s",argv[0],argv[1]);  printf("\nHello, %s\nWelcome to the world of programming\n\n",argv[1]);  return 0;  }  Output:  user69@trainux01:~/day21$ ./a.out Tejaswi  ec2.c  PID: 26272  ./a.out  Tejaswi  Hello, Tejaswi  Welcome to the world of programming |

|  |
| --- |
| #include <stdio.h>  #include<stdlib.h>  #include<sys/types.h>  #include<unistd.h>  #include<sys/wait.h>  int main(int argc, char \*argv[]) {  printf("\nIn the sysPrg06.c\n")  //printf("\n%s\n%s",argv[1],argv[2]);  execl(argv[1],argv[2],(char\*)0);  printf("\nThis line will not print");  return 0;  }  Output: |

……………………………………………………………………………………………….

**WAP for the calculator using command line argument supply operation and operands**

**./exc1 ./calc ./calc +2 3 ---for running in same directory**

**/home2/user69/day21/exc1 /home2/user69/day21/exc**

|  |
| --- |
| #include <stdio.h>  #include <stdlib.h>  int main(int argc, char const \*argv[])  {  char op=argv[1][0];  int op1 = atoi(argv[2]);  int op2 = atoi(argv[3]);  switch(op)  {  case '+':  printf("\nAdd=%d",(op1+op2));  break;  case '-':  printf("\nSub=%d",(op1-op2));  break;  }  printf("\nEnd of Program\n\n");  /\* code \*/  return 0;  }  Output :  **user69@trainux01:~/day21$ gcc calc.c -o calc**  **user69@trainux01:~/day21$ ./calc - 2 3**  **Sub=-1**  **End of Program**  **user69@trainux01:~/day21$ ./calc + 2 3**  **Add=5**  **End of Program** |

………………………………………………………………………………………………

**Signal:**

* **signal - ANSI C signal handling**

**SYNOPSIS**

#include <signal.h>

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

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

The behavior of signal() varies across UNIX versions, and has also varied historically across different versions of Linux. Avoid its use sigaction(2) instead. See Portability below signal() sets the disposition of the signal signum to handler, which is

either SIG\_IGN, SIG\_DFL, or the address of a programmer-defined func‐

tion (a "signal handler").

If the signal signum is delivered to the process, then one of the fol‐

lowing happens:

\* If the disposition is set to SIG\_IGN, then the signal is ignored.

\* If the disposition is set to SIG\_DFL, then the default action asso‐

ciated with the signal (see signal(7)) occurs.

\* If the disposition is set to a function, then first either the dis‐

position is reset to SIG\_DFL, or the signal is blocked (see Porta‐

bility below), and then handler is called with argument signum. If

invocation of the handler caused the signal to be blocked, then the

signal is unblocked upon return from the handler.

The signals SIGKILL and SIGSTOP cannot be caught or ignored.

……………………………………………………………………………………………….

Write a menu-based program to calculate a area of circle, rectangle. While u displayed the menu if a signal is interrupted u need to handle it.

CTRL+ c it should display the menu

U cannot terminate the program, if and only if u ended up the program.

It should display the cleaning of the program and then exit