Signals in UNIX

1)

//Use of sigint (^C)

#include <signal.h>

#include <stdio.h>

#include <unistd.h>

void oh(int sig)

{

printf("OH! - I got signal %d\n", sig);

signal(SIGINT,oh);/\* THIS LINE WILL CONTINUE THE EXECUTION OF FUNCTION OH \*/

//signal(SIGINT,SIG\_DFL);

}

int main()

{

signal(SIGINT, oh);

while(1)

{

printf("Hello World!\n");

sleep(1);

}

}

--------------------

2)

//Use of SIG\_DFL (reset to default)

#include <signal.h>

#include <stdio.h>

#include <unistd.h>

void oh(int sig)

{

printf("OH! - I got signal %d\n", sig);

signal(SIGINT,SIG\_DFL);/\*resets the signal to default\*/

}

int main()

{

signal(SIGINT, oh);

while(1)

{

printf("Hello World!\n");

sleep(1);

}

}

-------------------

3)

//Use of SIGQUIT (^C to quit process)

#include <signal.h>

#include <stdio.h>

#include <unistd.h>

void oh(int sig)

{

printf("OH! - I got signal %d\n", sig);

signal(SIGINT,oh);/\* THIS LINE WILL CONTINUE THE EXECUTION OF FUNCTION OH \*/

//signal(SIGQUIT,SIG\_DFL);

}

int main()

{

signal(SIGQUIT, oh);

while(1)

{

printf("Hello World!\n");

sleep(1);

}

}

-------------------

4)

//Use of SIGKILL (kill parent from child)

#include <stdio.h>

#include <stdlib.h>

#include <unistd.h>

#include <signal.h>

int main(void)

{

pid\_t ppid,pid,cpid;

ppid=getpid();

pid = fork();

if(ppid==getpid())

printf("parent");

else if (cpid==getpid())

printf("child");

if(pid > 0)

{

int i = 0;

while(i++ < 5)

{

printf("In the parent process.\n");

sleep(1);

}

}

else if (pid == 0)

{

int i = 0;

while(i++ < 10)

{

printf("In the child process.\n");

sleep(1);

if(i==3)

{

kill(ppid,SIGKILL); /\* SIGKILL Kills the process ( it cannot be caught or ignored)\*/

printf("Parent killed. I'm orphan!!!\n");

}

}

}

else

{

//something bad happened.

printf("Something bad happened.");

exit(EXIT\_FAILURE);

}

return 0;

}