Ctrl + C interrupt event handling in Linux

I am developing an application that uses C++ and compiles using Linux GNU C Compiler. However, I want to invoke a function as the user interrupts the script using Ctrl C keys. What should I do? Any answers would be much appreciated.

c++ c linux signals

edited Oct 17 '13 at 13:38
Grijesh Chauhan
39.5k 9 82 135

asked Jul 20 '13 at 20:47
mozcelikors
475 3 11 31

You need Defining Signal Handlers for SIGINT, get an idea from this code – Grijesh Chauhan Jul 20 '13 at 20:49

8 How does this have anything to do with the compiler used? – user529758 Jul 20 '13 at 20:52

That I need to use only libraries recognized by GCC – mozcelikors Jul 20 '13 at 20:54

3 @mozcelikors There's no such thing as "libraries recognized by GCC". If there's a library written in standard C, any sane C compiler should be able to compile it. – user529758 Jul 20 '13 at 20:55

1 possible duplicate of Handling Ctrl-C in a Linux TCP/IP Server written in C – user2485710 Jul 20 '13 at 21:04

3 Answers

When you press Ctr + C, the operating system sends a signal to the process. There are many signals and one of them is SIGINT. The SIGINT ("program interrupt") is one of the Termination Signals.

There are a few more kinds of Termination Signals, but the interesting thing about SIGINT is that it can be handled (caught) by your program. The default action of SIGINT is program termination. That is, if your program doesn't specifically handle this signal, when you press Ctr + C your program terminates as the default action.

To change the default action of a signal you have to register the signal to be caught. To register a signal in a C program (at least under POSIX systems) there are two functions

1. signal(int signum, sighandler_t handler);
2. sigaction(int signum, const struct sigaction *act, struct sigaction *oldact);.

These functions require the header signal.h to be included in your C code. I have provide a simple example of the `signal` function below with comments.

```
#include <stdio.h>
#include <stdlib.h>
#include <signal.h> //  our new library
volatile sig_atomic_t flag = 0;
void my_function(int sig){ // can be called asynchronously
  flag = 1; // set flag
}

int main(){
  // Register signals
  signal(SIGINT, my_function);
  //      ^          ^
  //  Which-Signal   |-- which user defined function registered
  while(1)
    if(flag){ // my action when signal set it 1
        printf("\n Signal caught!\n");
        printf("\n default action it not termination!\n");
        flag = 0;
    }
  return 0;
}
```

Note: you should only call safe/authorized functions in signal handler. For example avoid calling printf in signal handler.

You can compile this code with gcc and execute it from the shell. There is an infinite loop in the code and it will run until you send a `SIGINT` signal by pressing Ctr + C.

edited May 23 at 12:18
Community ♦
1 1

answered Jul 20 '13 at 21:42
Grijesh Chauhan
39.5k 9 82 135

You can leave comments for you doubts, I just try to give you a quick point of start. I believe with some more efforts you will explore the things in detail. – Grijesh Chauhan Jul 20 '13 at 21:53

1 Thanks for your kind replies and answer, that works great. – mozcelikors Jul 20 '13 at 22:03

I'm pretty sure the process is sent a signal, not a single. :-) – celtschk Jul 20 '13 at 22:27

@Duck hey Duck thanks man! my English is very poor, I notice your edits carefully and came to know about my mistakes. I will improve it. Thanks! – Grijesh Chauhan Jul 21 '13 at 5:10

1 @ Grijesh Chauhan, You're wlecome but your English isn't poor. I just filled in some holes. – Duck Jul 21 '13 at 5:45

Typing Ctrl C normally causes the shell to send `SIGINT` to your program. Add a handler for that signal (via `signal(2)` or `sigaction(2)`), and you can do what you like when Ctrl C is pressed.

Alternately, if you only care about doing cleanup before your program exits, setting up an exit handler via `atexit(3)` might be more appropriate.

edited Jul 20 '13 at 21:07

answered Jul 20 '13 at 21:02
Carl Norum
155k 19 289 374

You can use the signal macro.

Here is an example of how to deal with it:

```
#include <signal.h>
#include <stdio.h>
void sigint(int a)
{
    printf("^C caught\n");
}
int main()
{
    signal(SIGINT, sigint);
    for (;;) {}
}
```

Sample output:

```
Ethans-MacBook-Pro:~ phyrrus9$ ./a.out
^C^C caught
^C^C caught
^C^C caught
^C^C caught
^C^C caught
```

answered Jul 20 '13 at 21:22
phyrrus9
1,234 4 21

3 avoid use printf in signal handler – Grijesh Chauhan Jul 20 '13 at 21:25

Simple as that. I use it to close i2c connection so the printf will be replaced with connection killing functions. Thanks really. – mozcelikors Jul 20 '13 at 21:38