Signals, Part 3: Raising signals

jakebailey edited this page on Dec 16, 2014 · 8 revisions

How do I send a signal to a process from the shell?

You already know one way to send a `SIG_INT` just type `CTRL-C` From the shell you can use `kill` (if you know the process id) and `killall` (if you know the process name)

```
# First let's use ps and grep to find the process we want to send a signal to
$ ps au | grep myprogram
angrave  4409   0.0  0.0  2434892    512 s004  R+    2:42PM   0:00.00 myprogram 1

#Send SIGINT signal to process 4409 (equivalent of `CTRL-C`)
$ kill -SIGINT 4409

#Send SIGKILL (terminate the process)
$ kill -SIGKILL 4409
$ kill -9 4409
```

`killall` is similar except that it matches by program name. The next two example, sends a `SIGINT` and then `SIGKILL` to terminate the processes that are running `myprogram`

```
# Send SIGINT (SIGINT can be ignored)
$ killall -SIGINT myprogram

# SIGKILL (-9) cannot be ignored!
$ killall -9 myprogram
```

How do I send a signal to a process from the running C program?

Use `raise` or `kill`

```
int raise(int sig); // Send a signal to myself!
int kill(pid_t pid, int sig); // Send a signal to another process
```

For non-root processes, signals can only be sent to processes of the same user i.e. you cant just SIGKILL my processes! See kill(2) i.e. man -s2 for more details.

How do I send a signal to a specific thread?

Use `pthread_kill`

```
int pthread_kill(pthread_t thread, int sig)
```

In the example below, the newly created thread executing `func` will be interrupted by `SIGINT`

```
pthread_create(&tid, NULL, func, args);
pthread_kill(tid, SIGINT);
pthread_kill(pthread_self(), SIGKILL); // send SIGKILL to myself
```

Will `pthread_kill( threadid, SIGKILL)` kill the process or thread?

It will kill the entire process. Though individual threads can set a signal mask, the signal disposition (the table of handlers/action performed for each signal) is *per-process* not *per-thread*. This means `sigaction` can be called from any thread because you will be setting a signal handler for all threads in the process.

How do I catch (handle) a signal ?

You can choose a handle pending signals asynchronously or synchronously.

Install a signal handler to asynchronously handle signals use `sigaction` (or, for simple examples, `signal`).

To synchronously catch a pending signal use `sigwait` (which blocks until a signal is delivered) or `signalfd` (which also blocks and provides a file descriptor that can be `read()` to retrieve pending signals).

See `Signals, Part 4` for an example of using `sigwait`