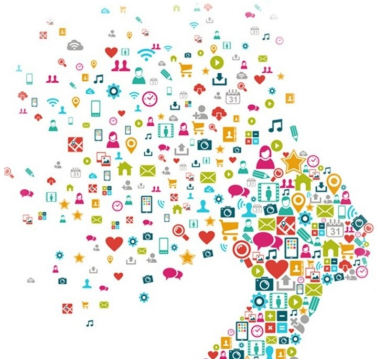


GLOBAL EDGE
Intelligence Of Things®

Signals

Abhijeet W

Contents

- What is signal ?
  - What is default action of signal ?
  - Data structure associated with signal
  - List of signals
- Signal Disposition
  - Signal system call & its drawback.
  - How to send signals via keystroke, command, function , system call ?
  - Sigaction system call & how to use it ?
- Sigevent

Contents (cnt ...)

- Some examples
- Summary
- Reference

What is signal ?

- A signal is a notification to a process that an event has occurred.

- Signals are two types
  - Traditional Signals or Standard Signals (1 to 31)
  - Real Time Signals

What is default action of signal ?

Default Action	Description
Term	Terminate process
Ing	Ignore signal
Core	Terminate process and dump core
Stop	Stop the process
Cont	Continue the process if process currently stopped

Data structure associated with signal

- Basic data structure used to store the signal sent to a process is a sigset_t array of bits.

```
typedef struct {
        unsigned long sig [2];
 }sigset_t;
```

- Maximum number of signal that may be declared in linux is 64.

List of standard signals

Table 9-1. The First 31 Signals in Linux/i386

#	Signal Name	Default Action	Comment	POSIX
1	SIGHUP	Abort	Hangup of controlling terminal or process	Yes
2	SIGINT	Abort	Interrupt from keyboard	Yes
3	SIGQUIT	Dump	Quit from keyboard	Yes
4	SIGILL	Dump	Illegal instruction	Yes
5	SIGTRAP	Dump	Breakpoint for debugging	No
6	SIGABRT	Dump	Abnormal termination	Yes
6	SIGIOT	Dump	Equivalent to SIGABRT	No
7	SIGBUS	Abort	Bus error	No
8	SIGFPE	Dump	Floating point exception	Yes
9	SIGKILL	Abort	Forced process termination	Yes
10	SIGUSR1	Abort	Available to processes	Yes
11	SIGSEGV	Dump	Invalid memory reference	Yes
12	SIGUSR2	Abort	Available to processes	Yes
13	SIGPIPE	Abort	Write to pipe with no readers	Yes
14	SIGALRM	Abort	Real timer clock	Yes
15	SIGTERM	Abort	Process termination	Yes
16	SIGSTKFLT	Abort	Coprocessor stack error	No
17	SIGCHLD	Ignore	Child process stopped or terminated	Yes
18	SIGCONT	Continue	Resume execution, if stopped	Yes
19	SIGSTOP	Stop	Stop process execution	Yes
20	SIGTSTP	Stop	Stop process issued from tty	Yes
21	SIGTTIN	Stop	Background process requires input	Yes
22	SIGTTOU	Stop	Background process requires output	Yes
23	SIGURG	Ignore	Urgent condition on socket	No
24	SIGXCPU	Abort	CPU time limit exceeded	No
25	SIGXFSZ	Abort	File size limit exceeded	No
26	SIGVTALRM	Abort	Virtual timer clock	No
27	SIGPROF	Abort	Profile timer clock	No
28	SIGWINCH	Ignore	Window resizing	No
29	SIGIO	Abort	I/O now possible	No
29	SIGPOLL	Abort	Equivalent to SIGIO	No
30	SIGPWR	Abort	Power supply failure	No
31	SIGUNUSED	Abort	Not used	No

Signal Dispositions

- UNIX systems provide two ways of changing the disposition of a signal: signal() and sigaction().

- Exception : A program can't handle signals of type SIGKILL, SIGSTOP.

Signal system call

- Signal () :-

  ANSI C signal handling.

- Synopsis :-

  #include <signal.h>

  typedef void (*sighandler_t)(int);

  sighandler_t signal(int signum, sighandler_t handler);

- Return Value :-

  On success signal() returns previous value of signal handler.

  On error SIG_ERR or EINVAL invalid signum.

Description

- The signal() system call a new signal handler for the signal with number signum.
- The signal handler is set to SIGhandler which may be a user specified function, or either SIG_IGN or SIG_DFL.

Drawback of signal system call

- Signal function change the new action then return the previous value.
- When SIGhandler is executing due to one signal but suddenly process get another signal comes kernel delivering another signal.
- Using signal function not possible to inform kernel block other signal when SIGhandler is executing due to one signal.

Sending signal via signal

- SIGINT : (ctrl-c)
  - Signal number : 2
  - Default action terminate the process.
- SIGTSTP : (ctrl-z)
  - Signal number : 20
  - Default action terminate process.
- SIGQUIT : (ctrl-\)
  - Signal Number : 3
  - Default action terminated the process and dump core.

Sending signal via commands

- Kill :-

  Send a signal to process.

- Synopsis :

  kill -[options] <pid> [...]

- Use -l or -L to list available signals. Particularly useful signals include HUP, INT, KILL, STOP, CONT, and 0.

- Example :

  kill -SIGINT 1234

Sending signal via function call

- Function raise() :-

  raise – send signal to caller.

- Synopsis :-

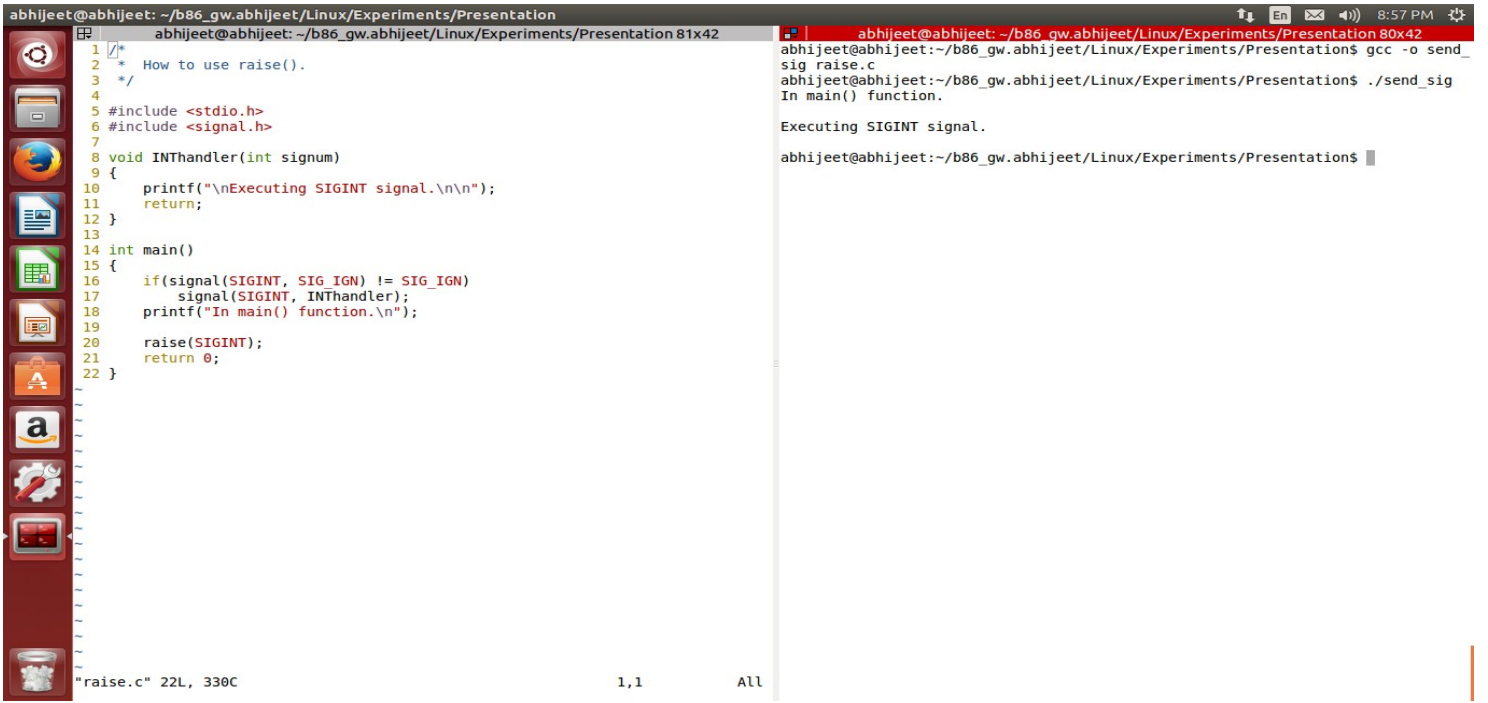
  #include <signal.h>

  int raise(int sig);

- Return Value :-

  raise() returns zero on success and nonzero on failure.

Sending signal via function call example



Sending signal via system call

- System call kill() :-

  Send signal to process.

- Synopsis :-

  #include <sys/types.h>

  #include <signal.h>

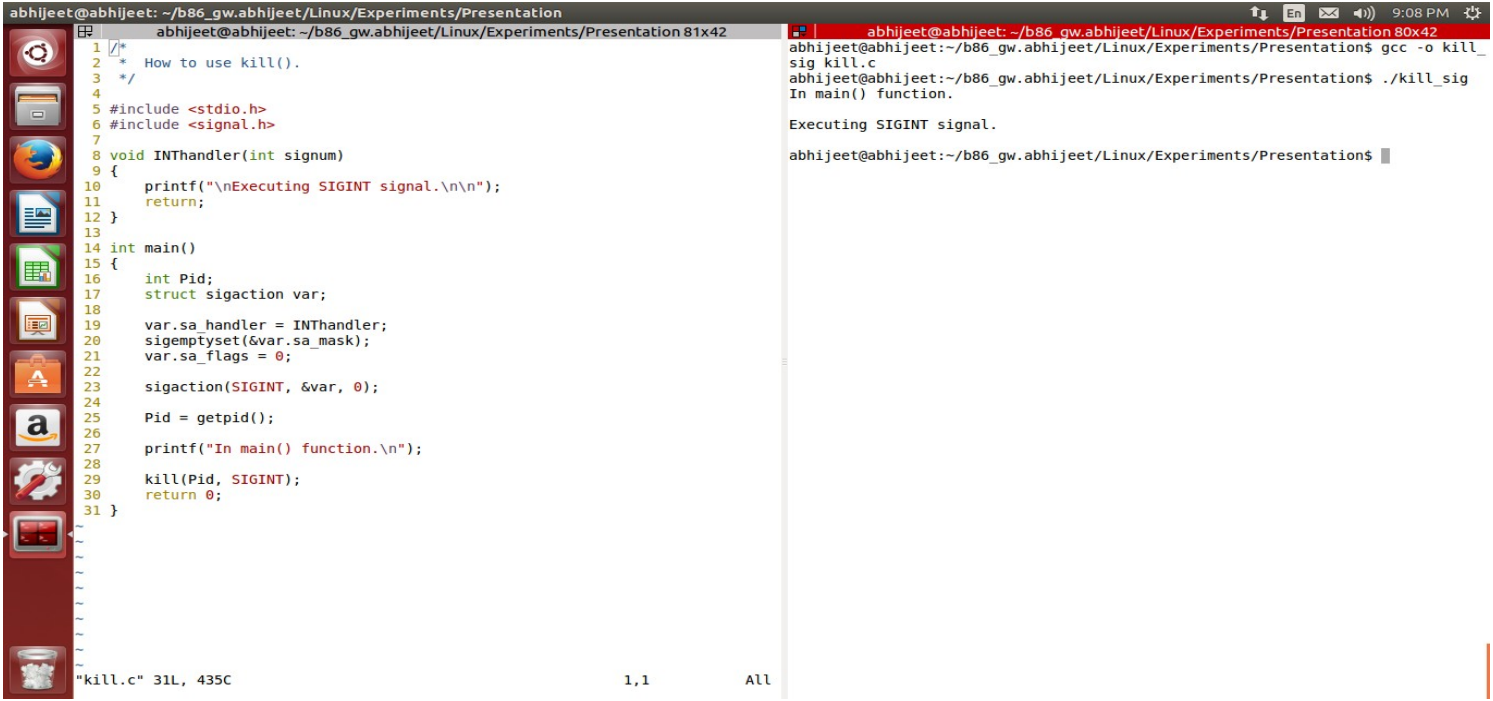
  int kill (pid_t pid, int sig);

- Return Value :-

  On success (at list one signal was sent) return zero.

  On failure -1 returned , and errno is set appropriately.

Sending signal via system call example



Sigaction system call

- Sigaction() :-

  Examine and change a signal action.

- Synopsis :-

  ```
  #include <signal.h>

  int sigaction(int signum, const struct sigaction *act, struct sigaction *oldact);
  ```

- The sigaction() system call is used to change the action taken by a process, on receiving of specific signal.

Sigaction system call (cnt..)

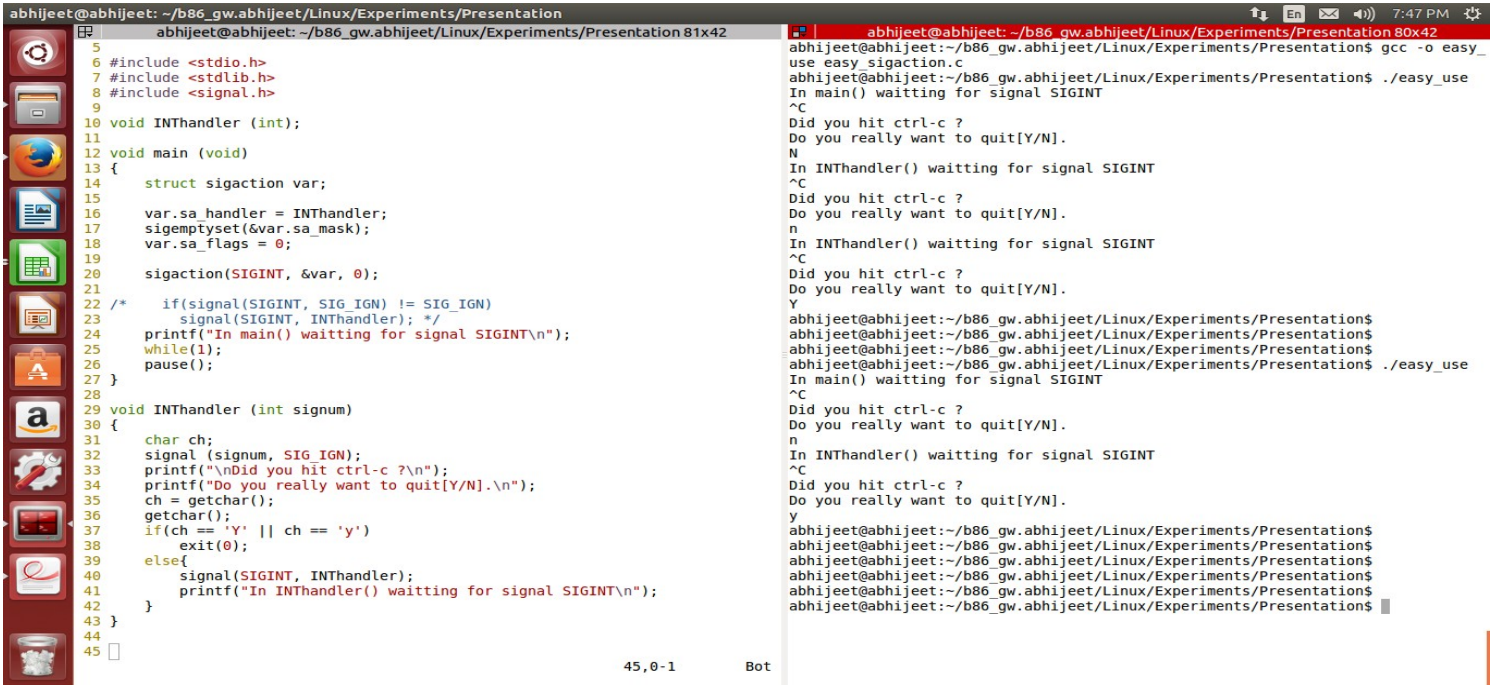
- Sigaction structure members :-

```
struct sigaction {
        void (*sa_handler) (int);
        void (*sa_sigaction) (int, siginfo_t *, void*);
        sigset_t sa_mask;
        int sa_flags;
        void (*sa_restorer) (void);
        };
```

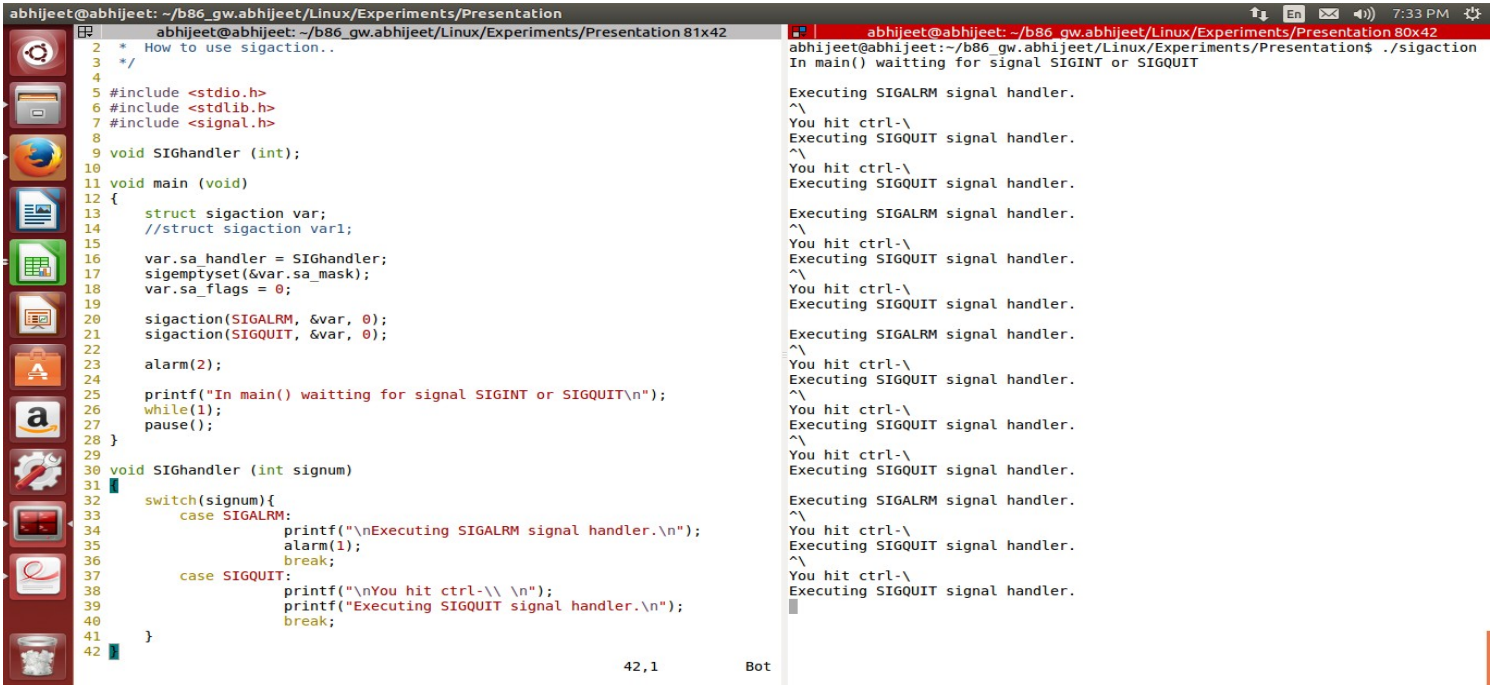
- Return value :-

sigaction() returns 0 on success ,on error returns -1.

How to use sigaction()



How to use sigaction() (cnt ...)



Sigevent

- Sigevent :-

  Structure for notification from asynchronous routines.

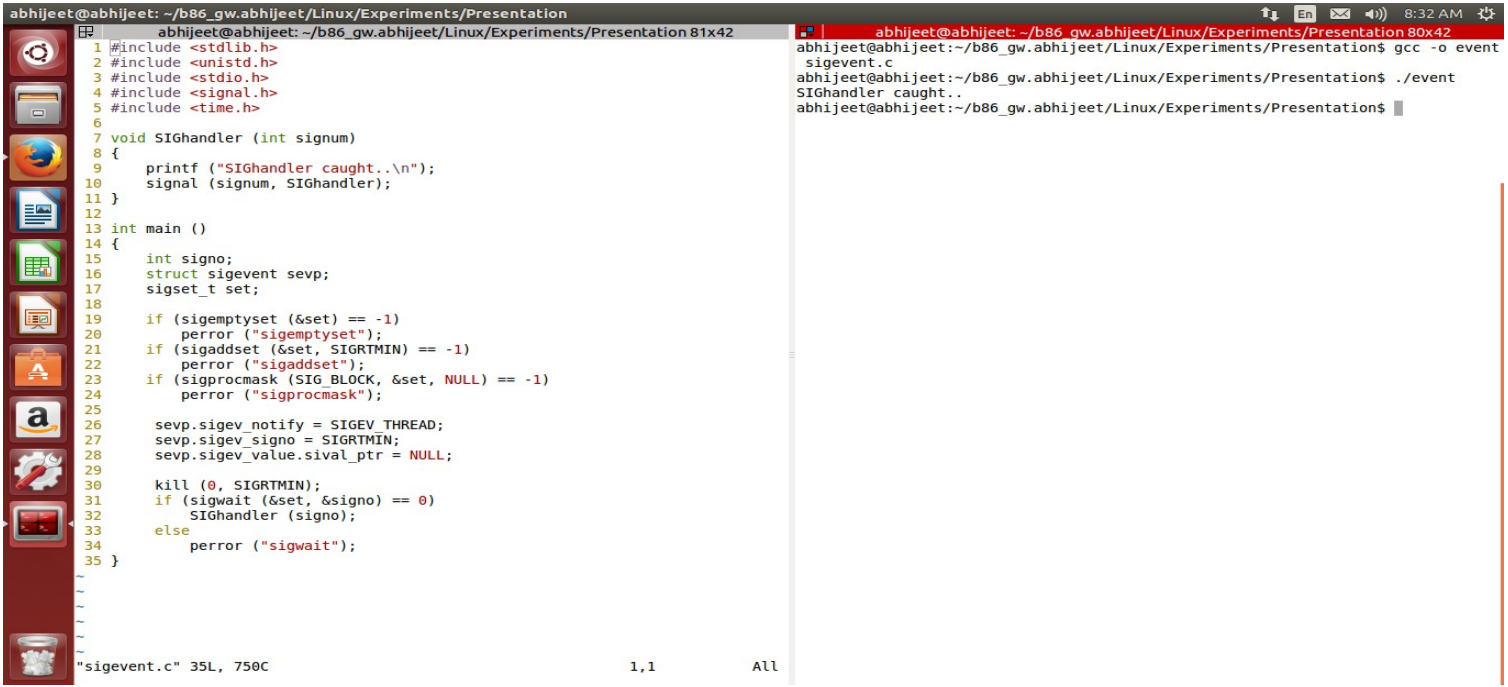
- Synopsis :-

```
union sigval {          /* Data passed with notification */
    int     sival_int;          /* Integer value */
    void   *sival_ptr;          /* Pointer value */
};

struct sigevent {
    int          sigev_notify; /* Notification method */
    int          sigev_signo;  /* Notification signal */
    union sigval sigev_value;  /* Data passed with
                                  notification */
    void       (*sigev_notify_function) (union sigval);
                    /* Function used for thread
                       notification (SIGEV_THREAD) */
    void        *sigev_notify_attributes;
                    /* Attributes for notification thread
                       (SIGEV_THREAD) */
    pid_t        sigev_notify_thread_id;
                    /* ID of thread to signal (SIGEV_THREAD_ID) */
};
```

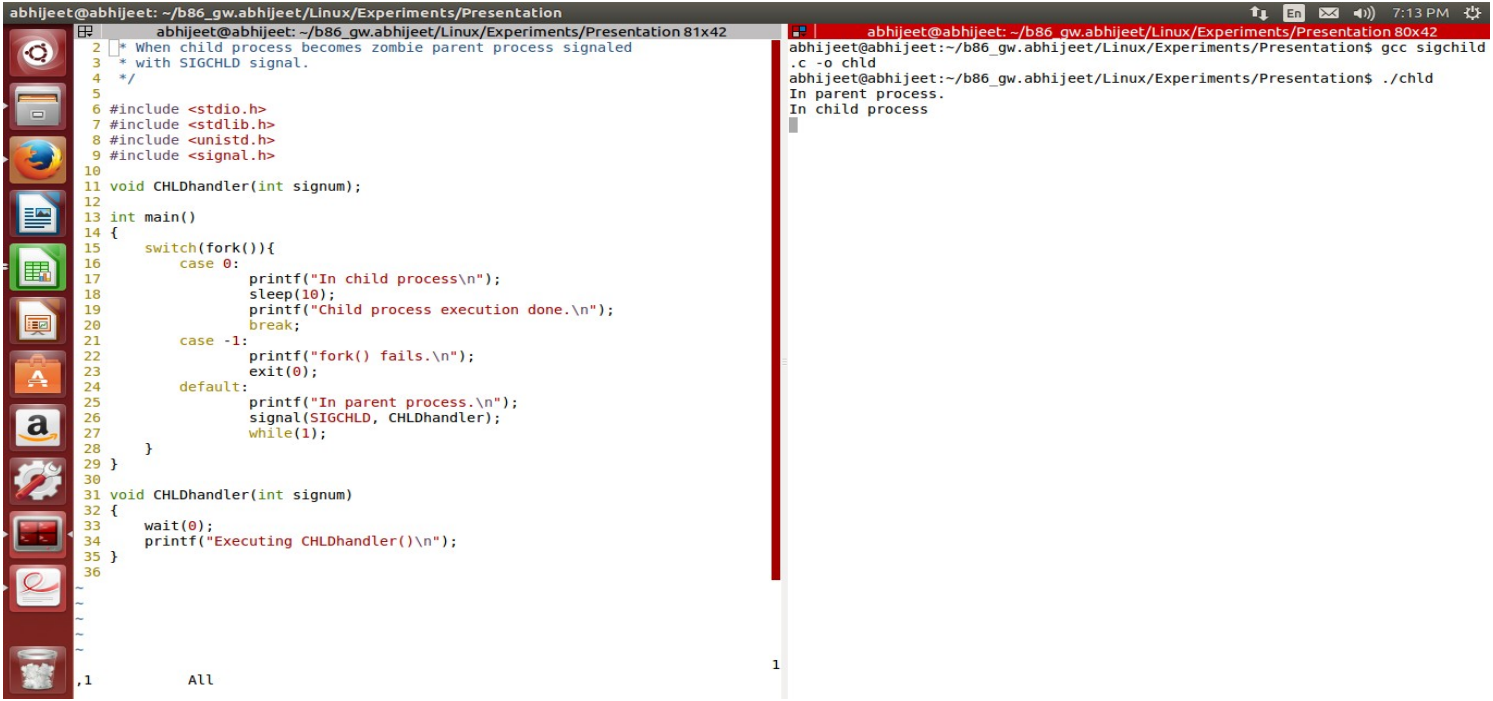
Sigevent (cnt ...)

- The sigevent structure is used by various APIs to describe the way a process is to be notified about an event. (e.g. Expiration of a timer, or the arrival of a message.)

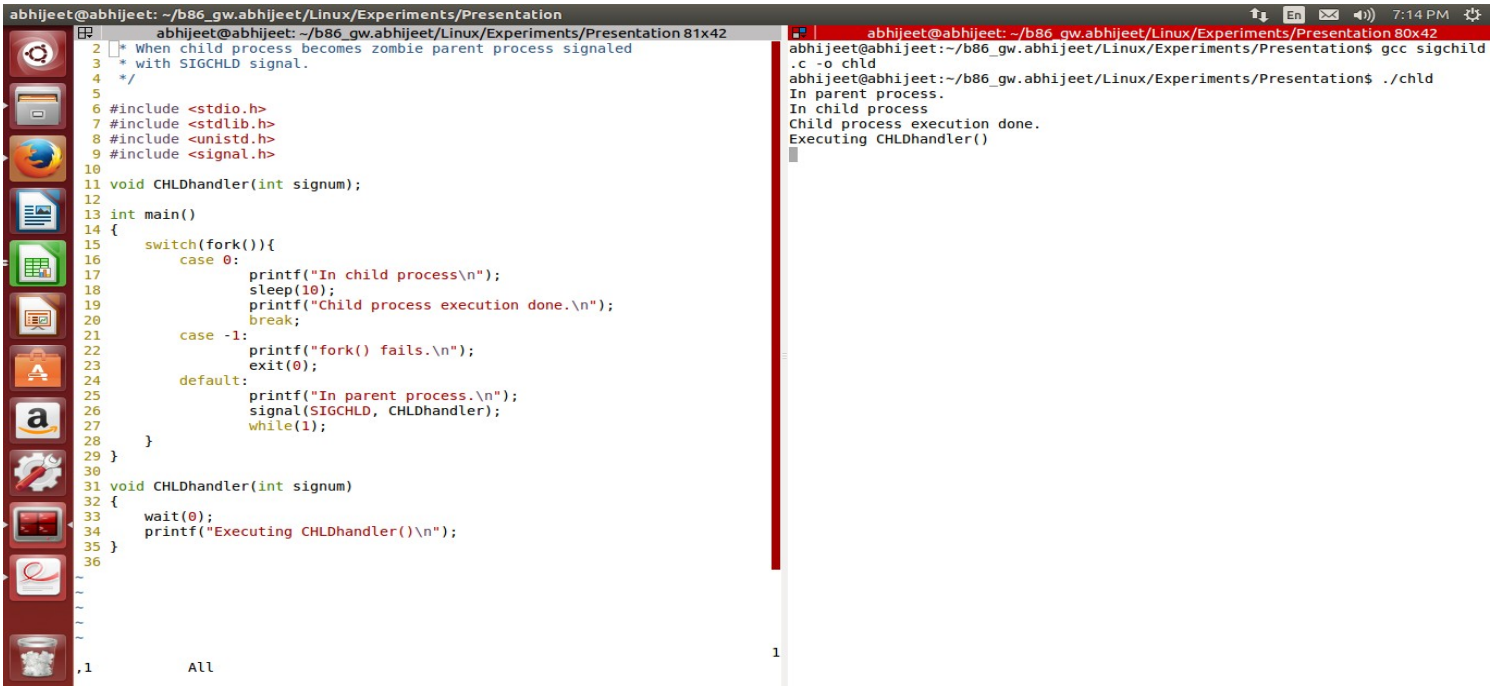
Sigevent Example



How to handle SIGCHLD signal



How to handle SIGCHLD signal (cnt...)



How to handle SIGSEGV signal.

abhijeet@abhijeet: ~/b86_gw.abhijeet/Linux/Experiments/Presentation

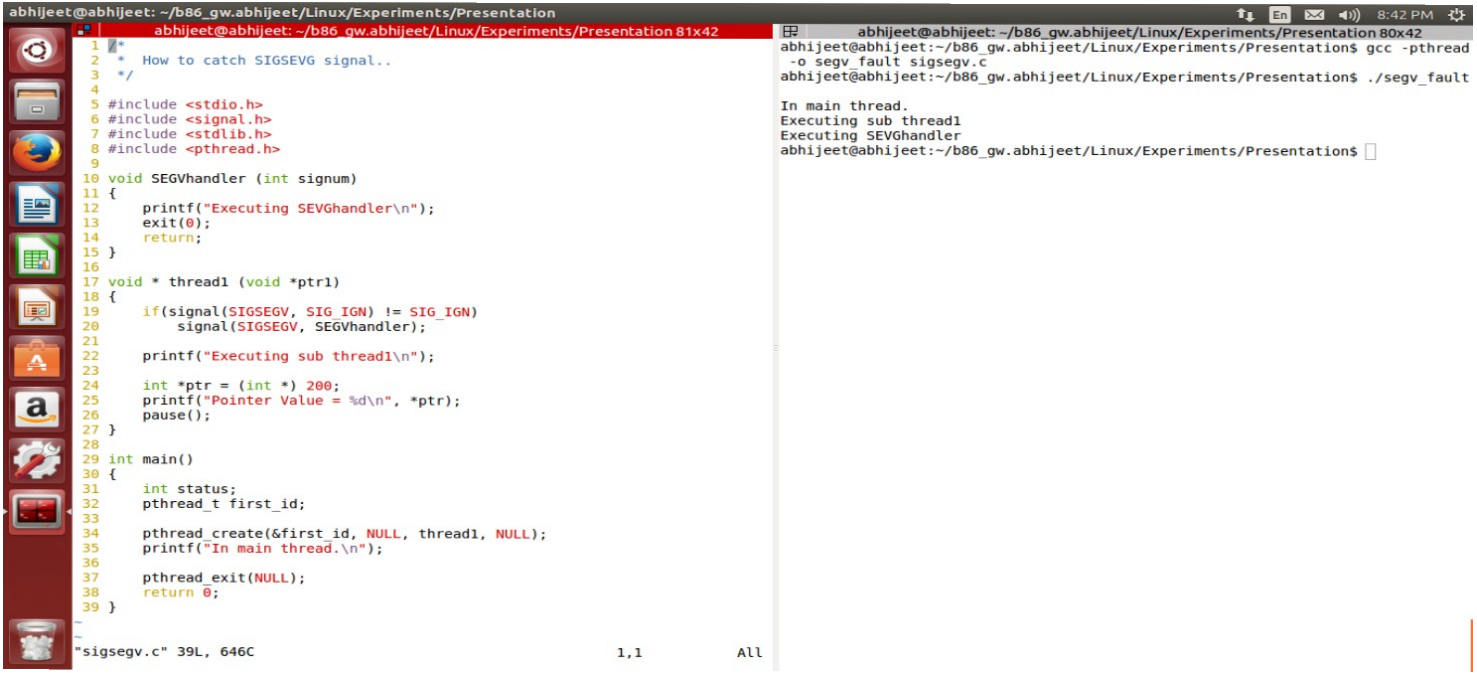
abhijeet@abhijeet: ~/b86_gw.abhijeet/Linux/Experiments/Presentation 81x42

```c
/*
 *  How to catch SIGSEVG signal..
 */

#include <stdio.h>
#include <signal.h>
#include <stdlib.h>
#include <pthread.h>

void SEGVhandler (int signum)
{
    printf("Executing SEVGhandler\n");
    return;
}

void * thread1 (void *ptr1)
{
    if(signal(SIGSEGV, SIG_IGN) != SIG_IGN)
        signal(SIGSEGV, SEGVhandler);

    printf("Executing sub thread1\n");

    int *ptr = (int *) 200;
    printf("Pointer Value = %d\n", *ptr);
    pause();
}

int main()
{
    int status;
    pthread_t first_id;

    pthread_create(&first_id, NULL, thread1, NULL);
    printf("In main thread.\n");

    pthread_exit(NULL);

    return 0;
}
~
~
"infi_sigsegv.c" 39L, 634C                    1,1           All
```

abhijeet@abhijeet: ~/b86_gw.abhijeet/Linux/Experiments/Presentation 80x42

```
Executing SEVGhandler
Executing SEVGhandler
Executing SEVGhandler
Executing SEVGhandler
Executing SEVGhandler
Executing SEVGhandler
Executing SEVGhandler
Executing SEVGhandler
Executing SEVGhandler
Executing SEVGhandler
Executing SEVGhandler
Executing SEVGhandler
Executing SEVGhandler
Executing SEVGhandler
Executing SEVGhandler
Executing SEVGhandler
Executing SEVGhandler
Executing SEVGhandler
Executing SEVGhandler
Executing SEVGhandler
Executing SEVGhandler
Executing SEVGhandler
Executing SEVGhandler
Executing SEVGhandler
Executing SEVGhandler
Executing SEVGhandler
Executing SEVGhandler
Executing SEVGhandler
Executing SEVGhandler
Executing SEVGhandler
Executing SEVGhandler
Executing SEVGhandler
Executing SEVGhandler
Executing SEVGhandler
Executing SEVGhandler
Executing SEVGhandler
Executing SEVGhandler
Executing SEVGhandler
Executing SEVGhandler
Executing SEVGhandler
Executing SEVGhandler
Executing SEVGha
```

How to handle SIGSEGV (cnt...)



Summary

In this presentation we discuss about basics of signal.

- What is signal , list of signals, default action of signals.
- Signal handling using signal() and sigaction().
- How create a event using sigevent.

References

- The Linux Programming Interface.

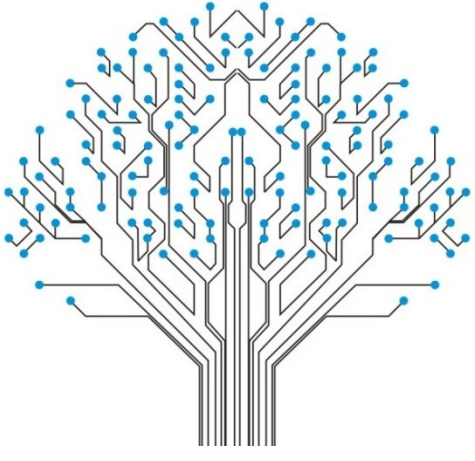
  By Michael Kerrisk.

- Understanding the Linux Kernel.

  By Daniel P. Bovet & Marco Cesati.

GLOBAL EDGE
Intelligence Of Things®

Thank you



Fairness

Learning

Responsibility

Innovation

Respect