Unix signal - higher level S/W form of ECF.
- allows processes and kernel to interrupt other processes.

Signal - small message that notifies a process than an event of some type has occurred in the system.

eg.
- SIGINT - interrupt from keyboard.
- SIGFPE - Floating point exception.
- SIGKILL - Kill Program
- SIGSEGV - Invalid memory reference
- SIGILL - Illegal Instruction.

[Diagram]
Kernel

SIGFPE
SIGILL
SIGSEGV

User Process

Ctrl + C

SIGINT

Process (foreground)
Signal Terminology

1. Sending a signal.

   ![Kernel ➔ Process]

   Reasons for sending a signal:
   i) Kernel has detected a system event
      eg. %0 by 0
      NULL pointer dereference.

   ii) Process A
       kill (pid, signal)

       ![Kernel ➔ Process B]
2. Receiving a signal

Proc A → signal → Kernel → signal → Proc B

Proc B is forced by the kernel to react to the signal in some way.

Proc B can either
1) Ignore the signal
2) Terminate
3) Catch the signal by executing a user-level function, i.e. "signal handler".

Process Groups
Each process belongs to exactly one process group.
Sending signals with `/bin/kill` program.

```
$ /bin/kill -9 15213
```

```
shell → SIGKILL (signal 9) → Process
        pid = 15213.
```

```
$ /bin/kill -9 -15213
```

```
shell → SIGKILL → pgid = 15213.
```

Sending signals from the **Keyboard** "Job" in a **UNIX** shell.

```
$ google-chrome & → background job.
```

```
$ ls | sort → foreground job.  
```

```
ls > pipe sort > o/p
```
Ctrl+C ➔ SIGHUP ➔ shell ➔ SIGINT ➔ Foreground job (FG)

Only one FG job at a time.
Default action for SIGINT ➔ Terminate the foreground job.

CTRL+Z ➔ SIGHUP ➔ shell ➔ Foreground process group

DEFAULT ACTION: STOP (Suspend) the foreground job.

Sending signals with the "kill" function

#include <sys/types.h>
#include <signal.h>

int kill(pid_t pid, int sig);

Returns: 0 if OK, -1 on error.
4. Sending signals with the "alarm" function.

```c
#include <signal.h>

unsigned int alarm (unsigned int secs);

Returns: Remaining secs of previous alarm, or 0 if no previous alarm.
```

- Process
  - `alarm (3)`
  - Sends an alarm to the kernel after 3 seconds.
  - Signals `SIGALRM` (after 3 seconds)

- Kernel
  - `alarm (2)`
  - Cancels any pending alarms.