Signals

- Notifications sent to a program by OS
  - Indicate special events
- Allows for asynchronous notification rather than polling
- Polling – to explicitly ask if something occurred, usually repeatedly

kill -l

SIGHUP	**SIGINT**	**SIGQUIT**	**SIGILL**	**SIGTRAP**
SIGABRT	**SIGBUS**	**SIGFPE**	**SIGKILL**	**SIGUSR1**
SIGSEGV	**SIGUSR2**	**SIGPIPE**	**SIGALRM**	**SIGTERM**
SIGCHLD	**SIGCONT**	**SIGSTOP**	**SIGTSTP**	**SIGTTIN**
SIGTTOU	**SIGURG**	**SIGXCPU**	**SIGXFSZ**	**SIGVTALRM**
SIGPROF	**SIGWINCH**	**SIGIO**	**SIGPWR**	**SIGSYS**
SIGRTMIN	SIGRTMIN+1	SIGRTMIN+2	SIGRTMIN+3	SIGRTMIN+4
SIGRTMIN+5	SIGRTMIN+6	SIGRTMIN+7	SIGRTMIN+8	SIGRTMIN+9
SIGRTMIN+10	SIGRTMIN+11	SIGRTMIN+12	SIGRTMIN+13	SIGRTMIN+14
SIGRTMIN+15	SIGRTMAX-14	SIGRTMAX-13	SIGRTMAX-12	SIGRTMAX-11
SIGRTMAX-10	SIGRTMAX-9	SIGRTMAX-8	SIGRTMAX-7	SIGRTMAX-6
SIGRTMAX-5	SIGRTMAX-4	SIGRTMAX-3	SIGRTMAX-2	SIGRTMAX-1
SIGRTMAX				

Common Error Signals

- **SIGILL** – Illegal Instruction
- **SIGBUS** – Bus Error, usually caused by bad data alignment or a bad address
- **SIGFPE** – Floating Point Exception
- **SIGSEGV** – Segmentation violation, i.e., a bad address

Termination Signals

- **SIGINT** – Interrupt, or what happens when you hit CTRL + C
- **SIGTERM** – Ask nicely for a program to end (can be caught)
- **SIGKILL** – Ask meanly for a program to end (cannot be caught)
- **SIGABRT**, **SIGQUIT** – End a program with a core dump

kill

- kill() is the system call that can send a process a signal (any signal, not just SIGKILL)

```c
#include <unistd.h>
#include <sys/types.h>
#include <signal.h>

int main() {
   pid_t my_pid = getpid();
   kill(my_pid, SIGSTOP);
   return 0;
}
```

kill

- From the shell in UNIX you can send signals to a program.
- Use ps to get a process ID

```
(1) thot $ ps -af
UID        PID  PPID  C STIME TTY          TIME CMD
jrmst106 27500 27470  0 07:13 ???      00:00:00 crashed_program
jrmst106 27507 27474  0 07:13 pts/5    00:00:00 ps -af
```

- kill it!

`kill 27500` – Sends SIGTERM

`kill -9 27500` – Sends SIGKILL

Catching Signals

- Some signals can be caught like exceptions in Java
- Do some cleanup, then exit
- Generally bad to try to continue, the machine might be in a corrupt state
- Some signals can be caught safely though

SIGALRM

```
#include <unistd.h>
#include <signal.h>

int timer = 10;

void catch_alarm(int sig_num) {
        printf("%d\n",timer--);
        alarm(1);
}

int main() {
        signal(SIGALRM, catch_alarm);

        alarm(1);
        while(timer > 0) ;
        alarm(0);
        return 0;
}
```

SIGTRAP

- Breakpoint trap
  - Debuggers listen for this
- OS Sends it when breakpoint trap instruction is hit
  - `int 3` on x86
- `int 3` is special (Why?)
  - 1 byte encoding: 0xCC
  - All other traps are two bytes: 0xCD 0x80 (linux syscall)