Signals

↳ Asynchronous notifications of external events

↳ come from

- Hardware
- software
- terminal line discipline

↳ when they happen

- current program is interrupted
- action
  - ↳ ignored
    - ↳ default action
    - ↳ catch :)
- after / during
  - ↳ action is reset to default (or not)
  - ↳ further delivery of signal is blocked (or not)
  - ↳ "long" syscalls interrupted (or not)

SIGSEGV
SIGILL
SIGPWR
} Hardware

SIGINT
SIGTSTP
SIGSTOP
SIGCONT
SIGTERM
SIGKILL
SIGWINCH

int kill(pid_t pid, int signal);
int raise(int signal)
↳ kill(getpid(), signal);

void (*signal(int signum, void (*handler)(int)))(int);

typedef void (*sigfun)(int);

sigfun signal(int signum, sigfun handler);
↳ old value or SIG_ERR
↳ 1 of SIG_IGN ← ignore
SIG_DFL ← default
pointer to a function

```
    x = x + 1;

int main(int argc, char *argv[]){
    if(signal(SIGINT, handler) == SIG_ERR){
        perror("signal(");
        exit(3);
    }
    for(;;)
        pause();

    return 0;
}

void handler(int signum){
    signal(SIGINT, handler);

    printf("neener neener\n");
}
```