

Signal

Praktikum Sistem Operasi

Ilmu Komputer IPB

2023

Sinyal

- ▶ Bentuk komunikasi antarproses (IPC) yang paling sederhana.
- ▶ Contoh IPC yang lain¹:
 - ▶ *pipe*
 - ▶ *shared memory*
 - ▶ *message passing*
 - ▶ *socket*

¹Silberschatz *et al.* (2013), *Operating System Concepts*, hlm 130–147.

inter-process communication (on Linux)

by Julia Evans
@b0rk



in no particular order:

① Write it to a file

② Send it over the local network

(with a HTTP request or something)

③ Use a pipe!

'program1 | program2'

cool thing: you get buffering automatically

cool thing: you can easily switch to having the 2 programs on different machines.

④ shared memory

processes can share memory, not just threads on the same process!

see `shm_open`

cool thing: very fast/powerful (+scary!)

④ Unix domain sockets

Another way to send a stream of data.

cool thing: you can send file descriptors over a unix domain socket

Jenis sinyal

- ▶ Ada 31 jenis sinyal standar².
- ▶ Berikut sinyal yang dapat dikirim oleh *user* ke proses:
 - ▶ Ctrl-C: sinyal *interrupt* (SIGINT)
 - ▶ Ctrl-Z: sinyal *stop* (SIGTSTP), bisa dilanjutkan lagi
 - ▶ Ctrl-\: sinyal *quit* (SIGQUIT)

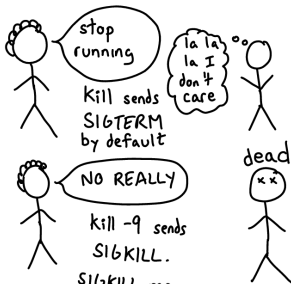
²lihat file `/usr/include/bits/signal-generic.h`.

SIGNALS

on Unix

julia evans
@bork
jvns.ca

If you've ever used
kill you've used signals.



stop running

la la la I don't care

NO REALLY

kill -9 sends SIGKILL.
SIGKILL means business

dead

Every program has an action it does for every signal

- terminate!
- ignore!
- custom handler
- core dump

You can customize how you respond to most signals! Many programs do. But not SIGKILL.

When you get SIGKILL you're dead

and every signal has a default action

SIGTERM → terminate

SIGHUP → terminate

SIGINT → terminate

When you press `Ctrl+C`, it sends the process a SIGINT!

SIGWINCH → ignore

"window resize" signal

+ lots more

```
#define SIGHUP      1    // hangup
#define SIGINT      2    // interrupt
#define SIGQUIT     3    // quit
#define SIGILL      4    // illegal instruction
#define SIGTRAP     5    // trace trap
#define SIGABRT     6    // abort
#define SIGBUS      7    // bus error
#define SIGFPE      8    // floating-point exception
#define SIGKILL     9    // kill, unblockable
#define SIGUSR1     10   // user-defined signal 1
#define SIGSEGV     11   // segmentation violation
#define SIGUSR2     12   // user-defined signal 2
#define SIGPIPE     13   // broken pipe
#define SIGALRM     14   // alarm clock
#define SIGTERM     15   // termination
```

```
#define SIGSTKFLT    16 // stack fault
#define SIGCHLD      17 // child terminated/stopped
#define SIGCONT      18 // continue
#define SIGSTOP      19 // stop, unblockable
#define SIGTSTP      20 // terminal stop
#define SIGTTIN       21 // background read from tty
#define SIGTTOU       22 // background write to tty
#define SIGURG        23 // urgent data on socket
#define SIGXCPU       24 // CPU time limit exceeded
#define SIGXFSZ       25 // file size limit exceeded
#define SIGVTALRM     26 // virtual timer expired
#define SIGPROF       27 // profiling timer expired
#define SIGWINCH      28 // window size change
#define SIGPOLL       29 // pollable event occurred
#define SIGPWR        30 // power failure imminent
#define SIGSYS        31 // bad system call
```

signal()

Fungsi signal()

```
void signal(int signum, void function(int));
```

- ▶ Untuk menangani sinyal yang masuk³.
- ▶ Jika ada signum yang masuk, maka function akan dijalankan.

³lihat man 2 signal.

Contoh

```
void foo(int sig) {  
    printf("got signal %d\n", sig); // print signum  
    signal(SIGINT, SIG_DFL); // back to default  
}  
  
int main() {  
    signal(SIGINT, foo);  
    while (1) {  
        puts("hello");  
        sleep(1);  
    }  
}
```

Penjelasan

- ▶ Jalankan program, kemudian kirim SIGINT (tekan Ctrl-C).
- ▶ Karena ada SIGINT masuk, program memanggil fungsi foo.
- ▶ Kirim lagi SIGINT.
- ▶ Apa yang terjadi? Mengapa demikian?
- ▶ Apa maksud SIG_DFL?

kill()

Fungsi kill()

```
int kill(pid_t pid, int signum);
```

- Untuk mengirim sinyal `signum` ke proses `pid`⁴.

⁴lihat man 2 kill.

Contoh

```
int main()
{
    pid_t child = fork();
    if (child == 0) {
        while (1) {
            puts("child");
            sleep(1);
        }
    } else {
        sleep(5);
        kill(child, SIGTERM);    // terminate
    }
    return 0;
}
```

Penjelasan

- ▶ *Child* akan terus mencetak tiap 1 detik.
- ▶ Setelah 5 detik, *parent* mengirim SIGTERM ke *child*.
- ▶ *Child* akan berhenti karena mendapat SIGTERM dari *parent*.

pause()

Fungsi pause()

```
int pause(void);
```

- ▶ Untuk menunggu sampai ada sinyal yang masuk⁵.

⁵lihat 'man 2 pause'.

Contoh

```
void ding(int sig) { puts("ding!"); }

int main()
{
    if (fork() == 0) {
        sleep(5);
        kill(getppid(), SIGALRM);
    } else {
        signal(SIGALRM, ding);
        puts("waiting...");
        pause();
    }
    return 0;
}
```

Penjelasan

- ▶ *Parent* menunggu sinyal masuk.
- ▶ *Child* akan mengirim SIGALRM ke *parent* setelah 5 detik.
- ▶ Setelah SIGALRM masuk, *parent* memanggil fungsi `ding`.
- ▶ Apa yang terjadi jika *parent* tidak memanggil fungsi `pause()`?

Tugas

Kirim Sinyal

- ▶ Modifikasi program contoh (p14) pada bagian *parent*, sehingga *child* akan:
 - ▶ berjalan selama 4 detik, lalu
 - ▶ berhenti sementara selama 3 detik, lalu
 - ▶ lanjut lagi berjalan selama 2 detik, lalu
 - ▶ selesai
- ▶ Tunjukkan ke asprak untuk dinilai
- ▶ Kumpulkan hasil akhirnya ke LMS (*file* [NIM] .c)