4/27

Last Class

1. System Calls
2. Process → User Mode
   → Kernel Mode
3. Context Switching.

Signals (8.5).

↳ a signal is a message that is sent to a process to notify it of an event that occurred.

↳ The process is notified.

↳ Two things can happen

① Handle the signal.

② Default signal action happens.

30 different signal types.

① Divide By Zero → man signal

SIGFPE

② Illegal Memory SIGSEGV.

Not in book

Divide-by-zero —Exception→ Kernel Exception Handler

Kernel Exception Handler —sends SIGFPE→ Process

Process → Handle it / Ignore it

Signals → high_level a software form of exceptional control flow.

↳ as a form of Inter-Process Communication.

Another example → SIG INT

(keyboard interrupt)

Ctrl + C → terminate (default action)

Ctrl + Z → SIGSTOP

(suspends) ← default action

Sending a Signal

When ?

① Exceptional event

② A request can be made to kill a process. (can kill itself)

Receive a Signal

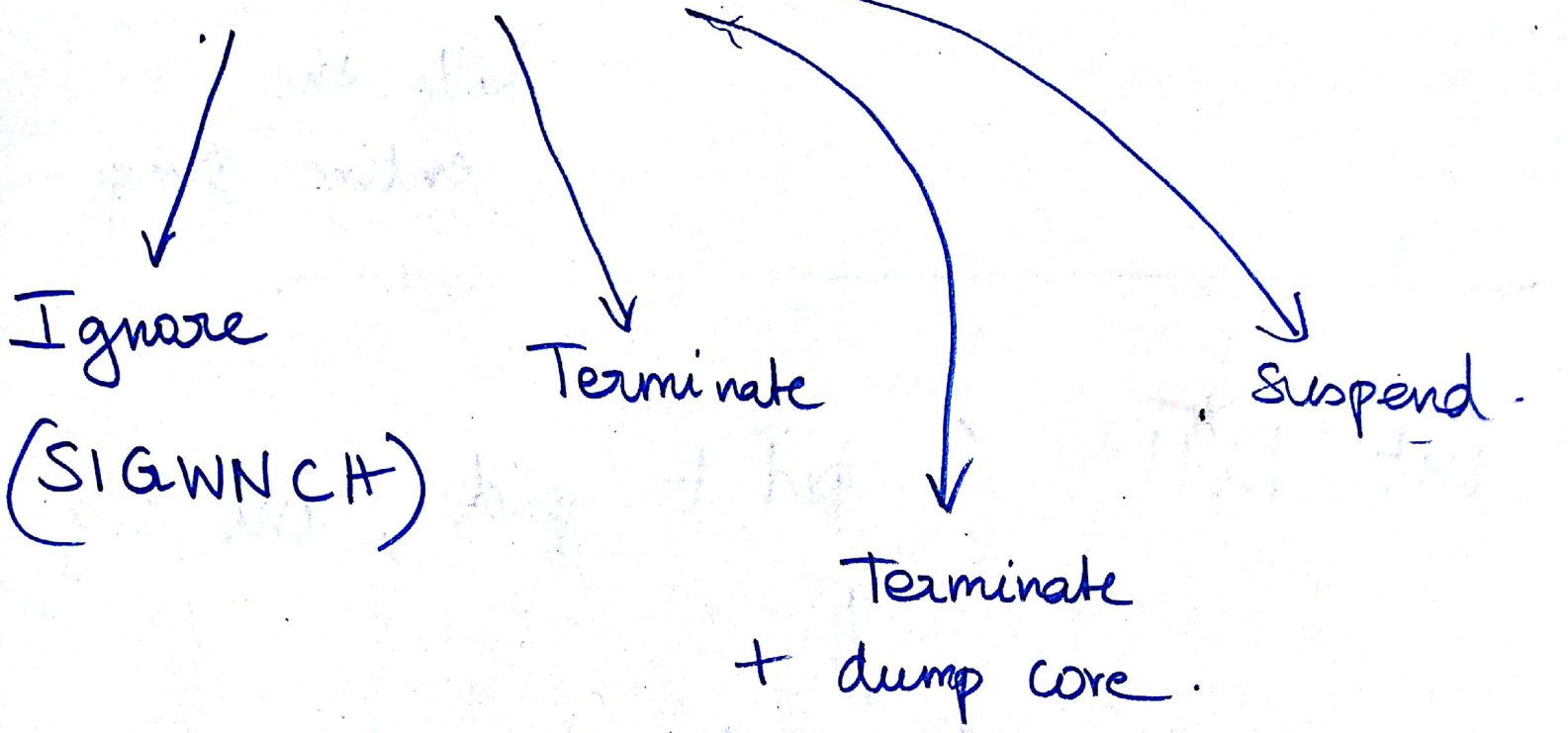
→ When a kernel is returning from an exception handler it checks if there are any pending events (queue).

→ If there are any (for process p), make the process p receive it.

On reception

(Ignore) When no handler

4 Default action



BUT → also catch a signal and handle it !

(some signals cannot be handled)

eg. SIGKILL.

Info → If two pending signals for a process p are the same, drop all of them except 1.

Block a signal

Stay in the pending queue till the proces unblocks it.

Example SIGKILL.

Use a utiliby. → /bin/ kill

```
$ /bin/kill -9 pid
        ↓
     SIGKILL
```

↓

help us kill a proces by sending SIGKILL.

Info → each proces belongs to a proces group

getpgrp() // setpgid(pid, pgid);

$ /bin/kill -9 -pid

↑ kills the entire group.

---

int kill (pid_t pid , int sig)

↓ pid of the process to be killed

↓ SIGKILL <signal.h>

Alarm function → SIGALRM.

Makes the kernel send a SIGALRM to the calling process.

alarm (unsigned int seconds).

We can handle it !

We can also handle multiple signals.

Signal Handling Info / Issues

① Slow system call → (read()).

interrupted by a signal ;

↪ Handle the signal. Once signal handler is done → What happens ?

(Typically, sys call returns an error).

② At most 1 pending signal of one type for each process.

If more come, drop!

③ If signal k is being handled, and a new signal k is sent to this process → it will be pending.

Signal of other type?

it interrupts the current signal handler.

BUT we can set it so that this doesn't happen.