#2 Algoritmes

1. Overzicht
2. Inleiding
3. Buffercache
    - `lru` Least Recently Used [p46]
    - `getblk` Lezen/Schrijven van diskblokken [p51]
    - `brelse` Vrijgave buffer [p54]
    - `bread` Block read [p63]
    - `breada` Block read & Read ahead [p63]
    - `bwirte` Block write [p64]
4. Files & INodes
    - `iget` Vind & Lock inode [p74]
    - `iput` Zet inde weer vrij, schrijf indien nodig [p76]
    - `bmap` Byte-offset -> Blocknr [p80]
    - `namei` path -> inode [p85]
    - `ialloc` Nieuwe inode zoeken [p90]
    - `ifree` inode vrijgeven/deleten [p92]
    - `alloc` Nieuwe block op disk zoeken [p93]
5. Sycalls voor FS
    - `open` Open file [p105]
    - `read` Lees N bytes uit file [p109]
    - `creat` Maak file [p120]
    - `mknod` Maak "speciale" files [p123]
    - `chdir` Verander cdir [p124]
    - `pipe` Maak een pipe [p127]
    - *skip `mount` [p138]*
    - *skip `iget` (met mount) [p141]*
    - *skip `namei` (met mount) [p141]*
    - *skip `umount` [p144]*
    - `link` *Hard*-Link file [p147]
    - `unlink` Verwijder (naam van) file [p151]
6. Proces structuren
    - `inthand` Handel interrupt/exception af [p188]
    - `syscall` Handel syscall af [p191]
    - context switch [p196]
    - *skip `allocreg` [p199]*
    - *skip `attachreg` [p200]*
    - *skip `growreg` [p202]*
    - *skip `loadreg` [p205]*
    - *skip `freereg` [p207]*
    - *skip `detachreg` [p207]*
    - *skip `dupreg` [p209]*
    - `sleep` [p213]
    - `wakeup` [p214]
7. Procesbesturing
    - `fork` *Ga heen en vermenigvuldigt u* [p232]
    - `issig` Ontvang signalen (+kindjes dood) [p235]
    - `psig` Handel signalen af [p236]
    - `exit` *Ge moogt naar huis gaan* [p247]
    - `wait` Wacht op de dood van een kind [p249]
    - `exec` Transmuteer in een ander proces [p253]
    - *skip `xalloc` [p260]*
    - `brk` Vraag meer geheugen [p267]
    - `start` Bootstrap PID 0/1 [p274]
    - `init` PID1 [p276]
8. Scheduling

    - `schedule_proces` Het eigenlijke context switchen [p289]
    - `klok` Klok interrupt [p304]
9. Geheugenbeheer
    - `malloc` (Niet exact hetzelfde als libc malloc!) [p318]
    - `swapper` PID0 [p326]
    - `vfault` Page fault handler (geldigheid) [p347]
    - `pfault` Page fault handler (protection) [p352]
10. IO
    - *skip `open` (voor device drivers) [p367]*
    - *skip `close` (voor device drivers) [p370]*
    - `terminal_write` [p387]
    - `terminal_read` [p390]
    - `login` [p398]

#2 Tabellen

Niet volledig, maar het belangrijkste (zeker voor de teken oefening) staat er op. Zie ook de `UnixMemoryLayout` PDF! Gebaseerd op p171-173

- Region table
    - Type (Data, Stack, Text)
    - Reference Count
    - Size
    - → I-Node (bij type Text)
- Process table
    - PID & PPID & PGR
    - Status (Zombie, Running, ...)
    - Alarm
    - Signal
    - Nice
    - Exit
    - 'Per Process Region Table' (3x → Region)
- User table
    - → Process
    - Effective & Real UID & GUID
    - Current & Parent & Root DIR (3x → I-Node)
    - File mask
    - Signal handlers: 32x → functions, or SIG_DFL (0), or SIG_IGN (1)
    - 'User File Descriptor Table' (~20x → File)
    - (Kernel stack)
- File table
    - Mode
    - Reference Count
    - Offset
    - → I-Node
- I-Node table
    - Permissions
    - Reference Count
    - Link Count

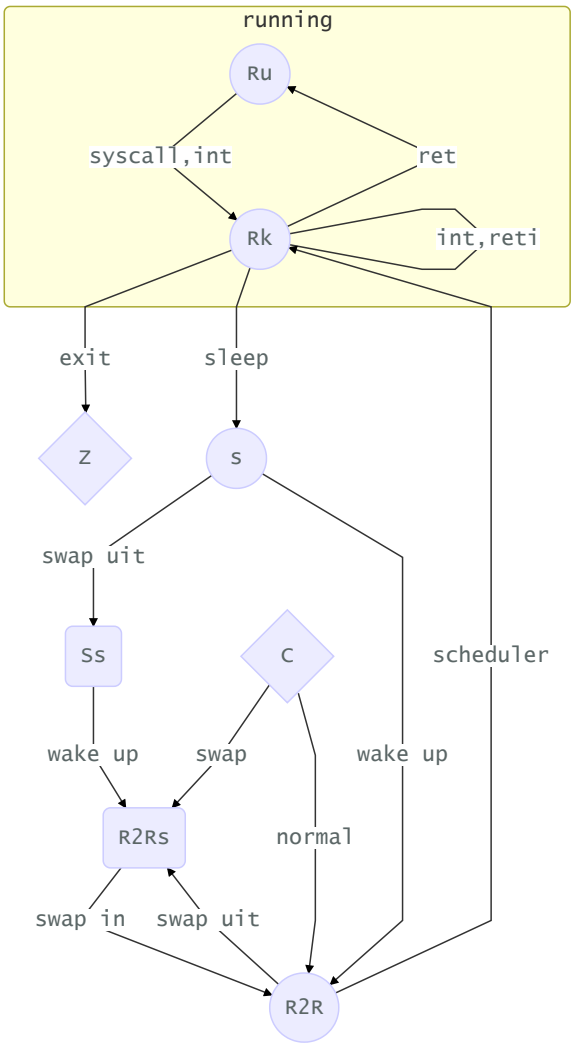
#2 Signals

Symbol	Number	Action	Meaning
SIGHUP	1	exit	Hangs up.
SIGINT	2	exit	Interrupts.
SIGQUIT	3	core dump	Quits.
SIGILL	4	core dump	Illegal instruction.
SIGTRAP	5	core dump	Trace trap.
SIGIOT	6	core dump	IOT instruction.
SIGEMT	7	core dump	MT instruction.
SIGFPE	8	core dump	Floating point exception.
SIGKILL	9	exit	Kills (cannot be caught or ignored).
SIGBUS	10	core dump	Bus error.
SIGSEGV	11	core dump	Segmentation violation.
SIGSYS	12	core dump	Bad argument to system call.
SIGPIPE	13	exit	Writes on a pipe with no one to read it.
SIGALRM	14	exit	Alarm clock.
SIGTERM	15	exit	Software termination signal.
SIGUSR1	16	exit	User defined signal 1
SIGUSR2	17	exit	User defined signal 2
SIGCHLD	18	ignore	Death of a child (special)
SIGPWR	19	ignore	Power fail

(Er zijn er 32, maar die laatste staan niet in de cursus.)

#2 Process states (Zie p169)

#	Afk.	Naam
1.	`Ru`	Running (user mode)
2.	`Ru`	Running (kernel mode)
3.	`R2R`	Ready to run
4.	`S`	Sleep
5.	`R2Rs`	Ready to run (in swap)
6.	`Ss`	Sleep (in swap)
7.	`I*`	Interrupted Bestaat eigenlijk niet, hier wordt proc onderbroken.
8.	`C`	Created
9.	`Z`	Zombie



Notities

"De kernel" is geen process, syscalls, interrupts, ... worden altijd uitgevoerd in de context vaan een user proccess. (Zie ook p186)
(Er zijn wel speciale processen, zoals swapper etc)

Interrupt (= verwacht) != exceptie (= onverwacht).
Excepties kunnen "in" een instructie komen (de instructie wordt dan opnieuw gedaan na de handler), interrupts enkel tussen twee instructies.
De afhandeling is echter wel hetzelfde (toch bij UNIX).

Kritieke zone = atomic operation (Moet in een keer, bvb lijst aanpassen).
Dan kunnen sommige interrupts worden uitgezet. Moet zo kort mogelijk zijn.

3. Buffercache

Everything's a file! (devices etc)
Block devices gaan via bufferchache, character devices niet.

Unix FS indeling:

```
Boot | Super | inodes ... | data ...
 512 |  512  |   i x 512  |  n x 512
```

- Boot block: Bootloader. Elke FS heeft die, maar 1 is nodig.
- Super block: Toestand van FS. (size, i, n, partial free lists, ...)
- Inodes: *Vaste lengte* inode lijst. 0 bestaat niet, 1 is de eerste.
- Data: De rest v/d disk, opgedeeld per block. Eke block kan maar tot 1 inode behoren.

Programma code = Text. (Er is ook Data en Stack.)

Er zijn 2 speciale processen, PID 0 en 1.
PID 0 wordt "met de hand" gemaakt bij boot. Die forkt naar PID 1.
PID 0 wordt de "swapper", PID 1 wordt "init".

Kernel stack is appart van user stack (staat in user structuur, fixed len)

Syscall gebeurd via `trap` instructie (software interrupt). Parameters via registers of stack.

Buffercache Status:

- Locked?
- Geldig?
- Delayed Write? (moet nog schijven voor vrijgeven)
- Oud? (deel van delayed write eigenlijk)
- Busy? (lezen of schrijven nu bezig)
- Proces waiting? (er wacht iemand op lock)

Buffercache voor/nadelen [p65]

4. Files & Inodes

Inode op disk heeft geen ref. count, want die zou altijd 0 zijn. [p72]

5. Sycalls voor FS

`dup` wijst naar hetzelfde filetabel element! Offset zijn dus shared!

`link` heeft *deadlock* scenario!

6. Proces structuren

`U` is enkel binnen huidige proc beschikbaar, daarom is dit apart van proc.

7. Procesbesturing

8. Scheduling

Prioriteit op basis van CPU usage [p293-296]

`nice` [p296]

UNIX is niet realtime. *niet 'heel heel snel' dus* :P

9. Geheugenbeheer

Swappen voor een fork [p323]

Demand paging [p331]

10. IO

Yet another cache: clijsten [p383]