Основные принципы и понятия

Операционные системы

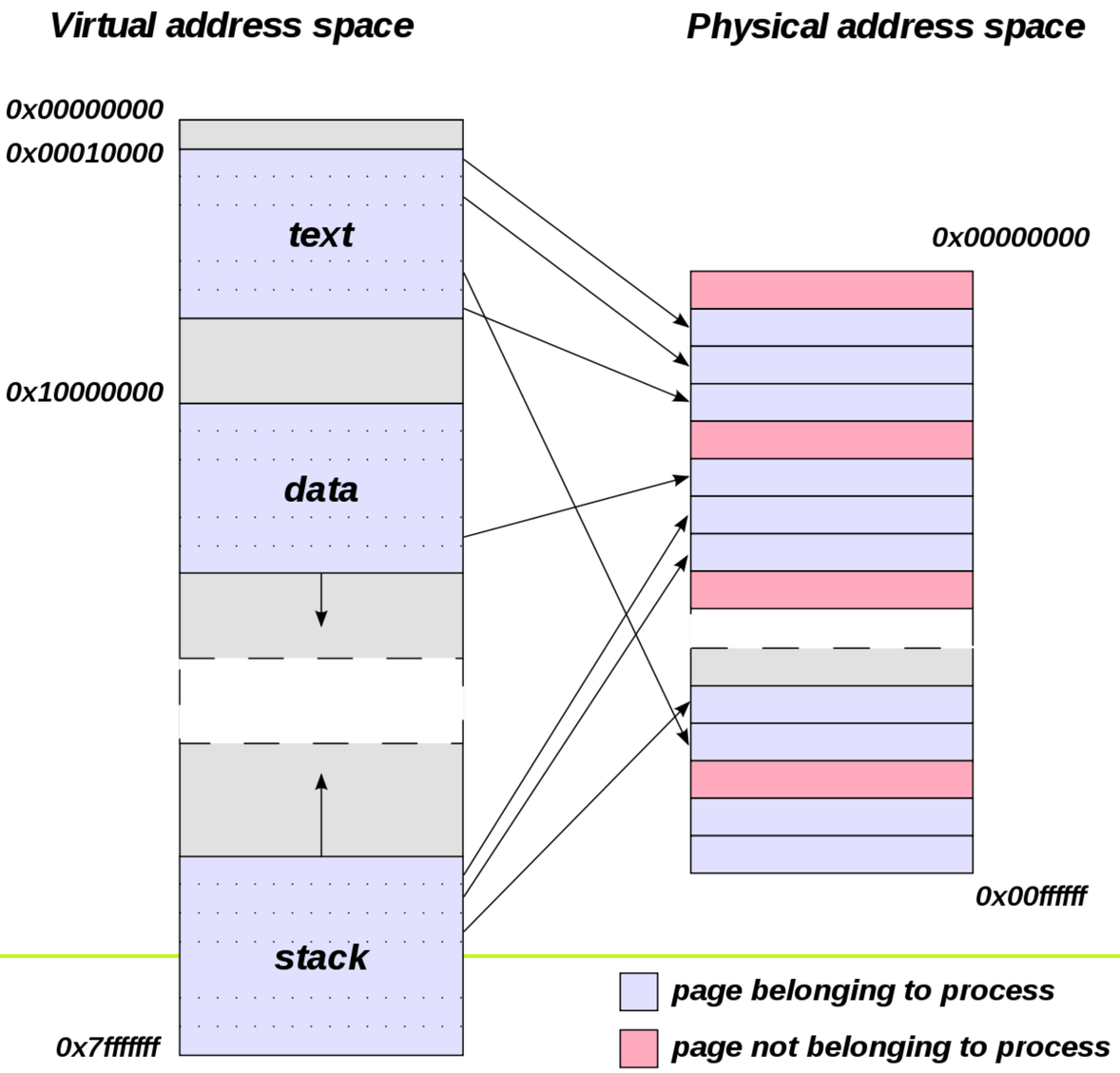
НИКОЛАЙ ИГОТТИ

Процессы

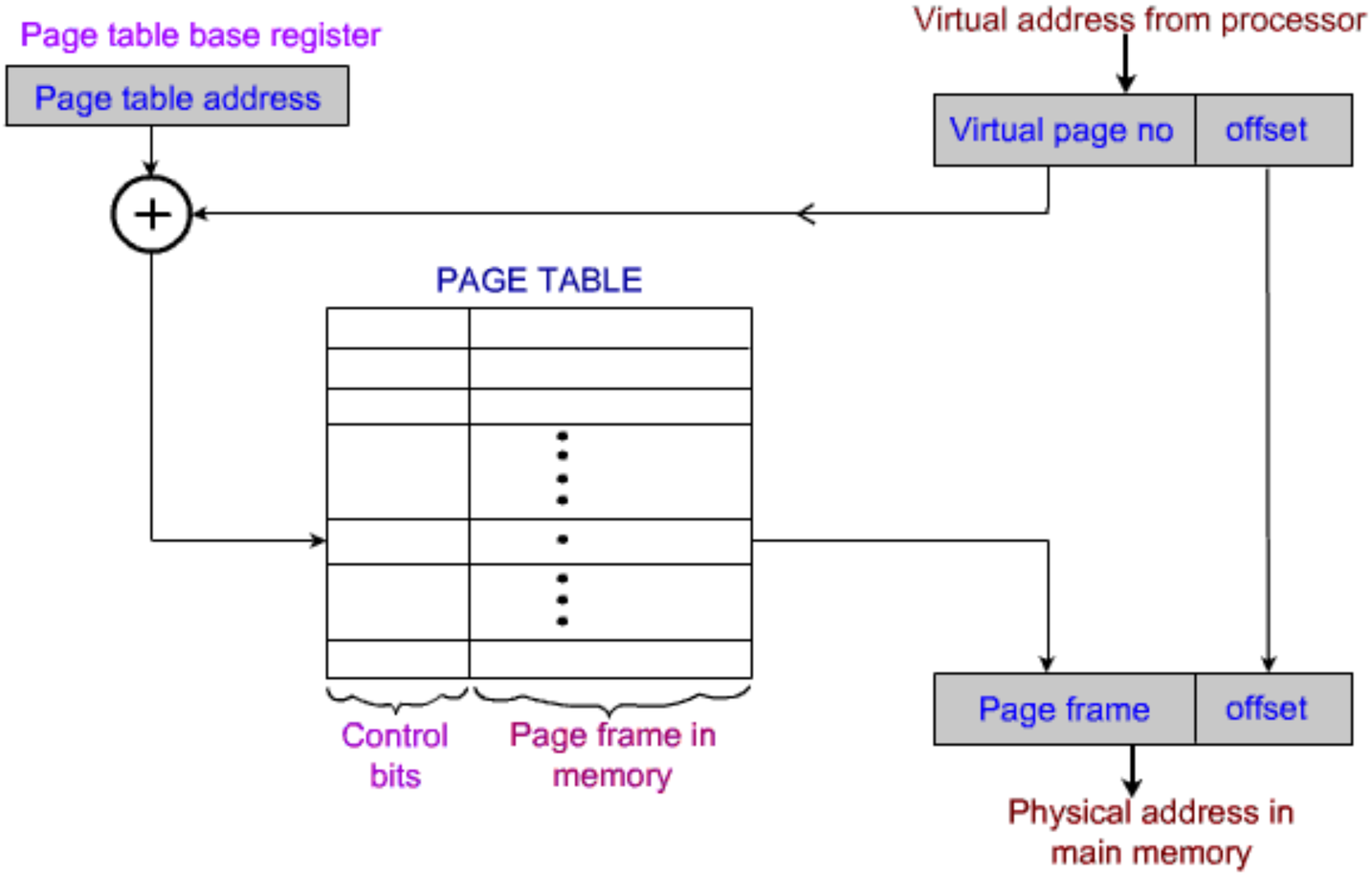
Процесс это…

- Адресное пространство с уникальной виртуальной памятью (x86: CR3, arm: TTBR0, TTBR1)
- Один или несколько потоков исполнения
- В непривилегированном контексте процессора (x86: ring 3, arm: User)
- С стандартизированным интерфейсом в ядро (x86: syscall/sysenter/int, arm: svc)
- С собственным списком ресурсов (файлы, handles, etc.)
- Уникальный (рециклируемый) идентификатор pid

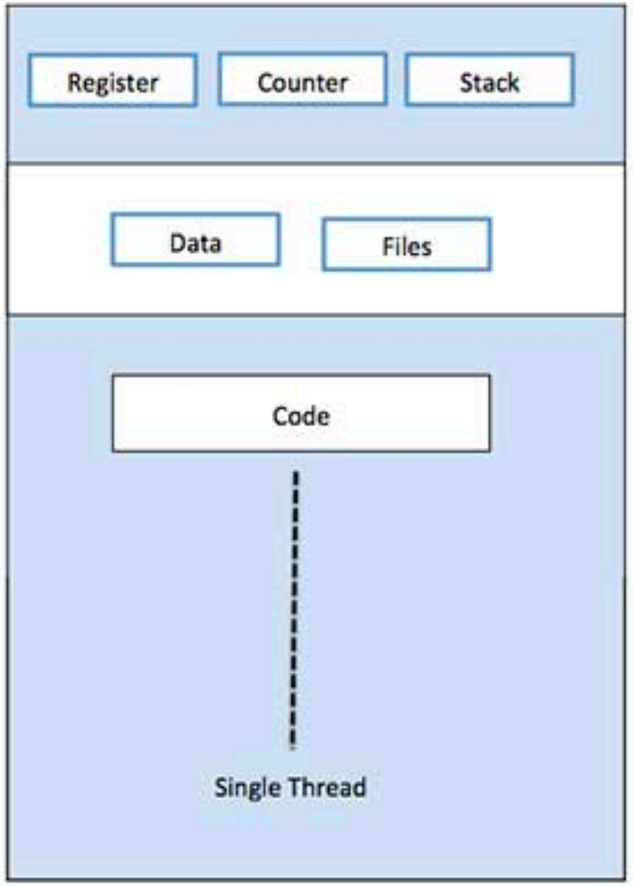
Адресное пространство



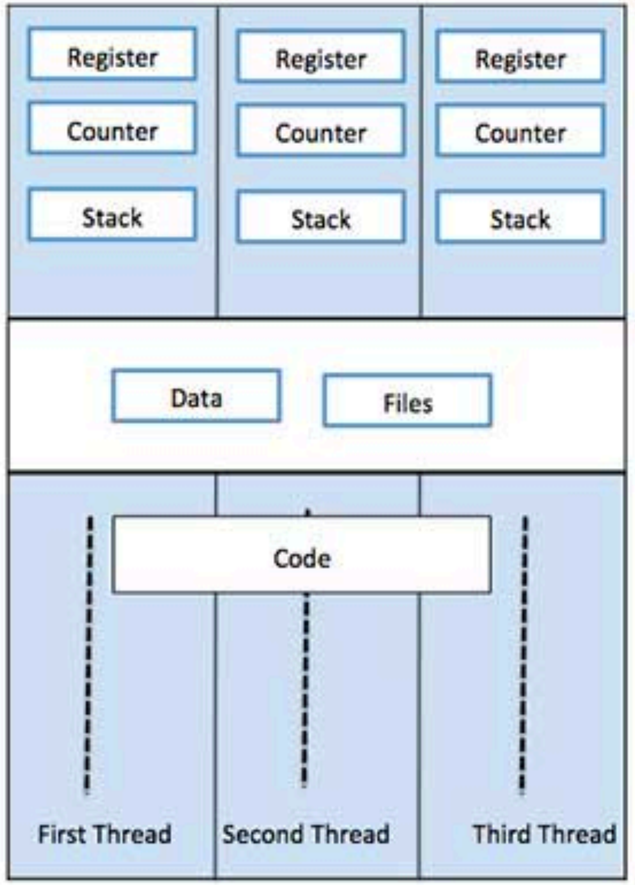
Адресное пространство, трансляция



Поток исполнения

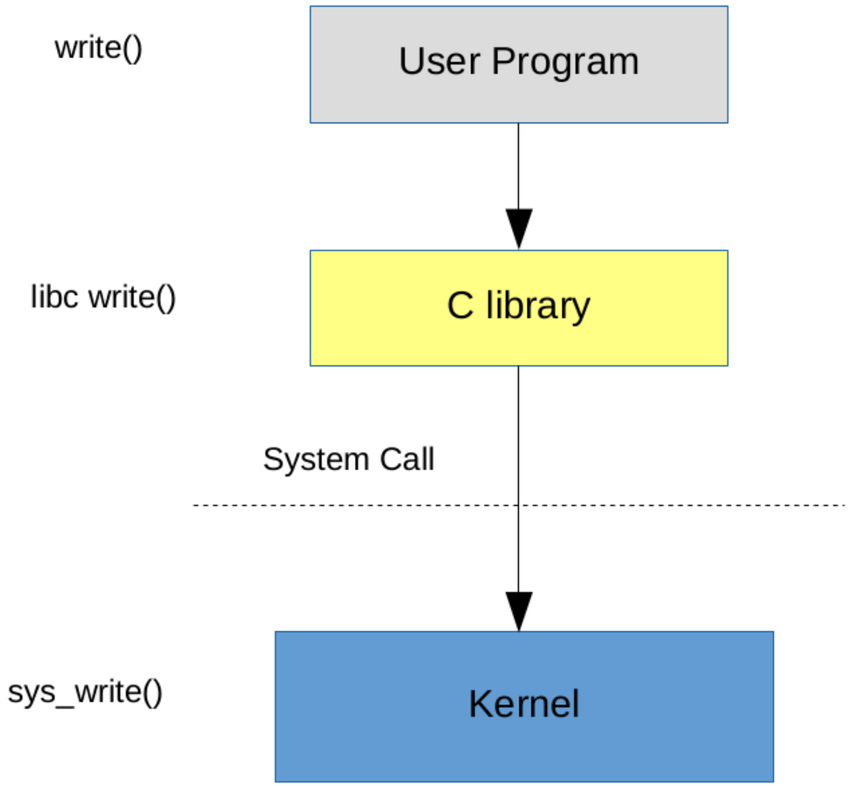


Single Process P with single thread



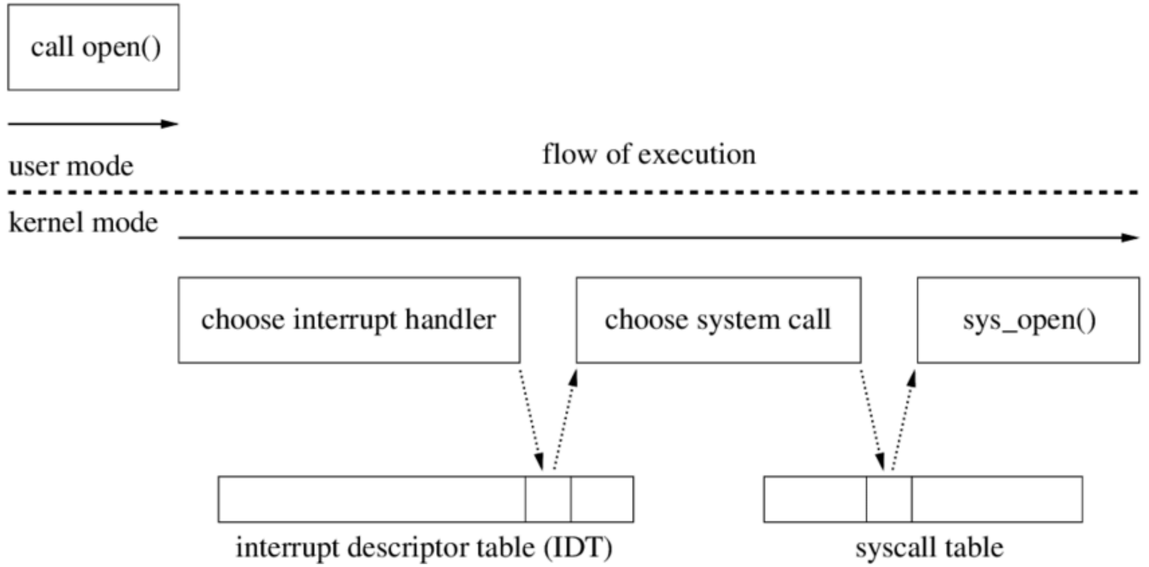
Single Process P with three threads

Системный вызов, syscall

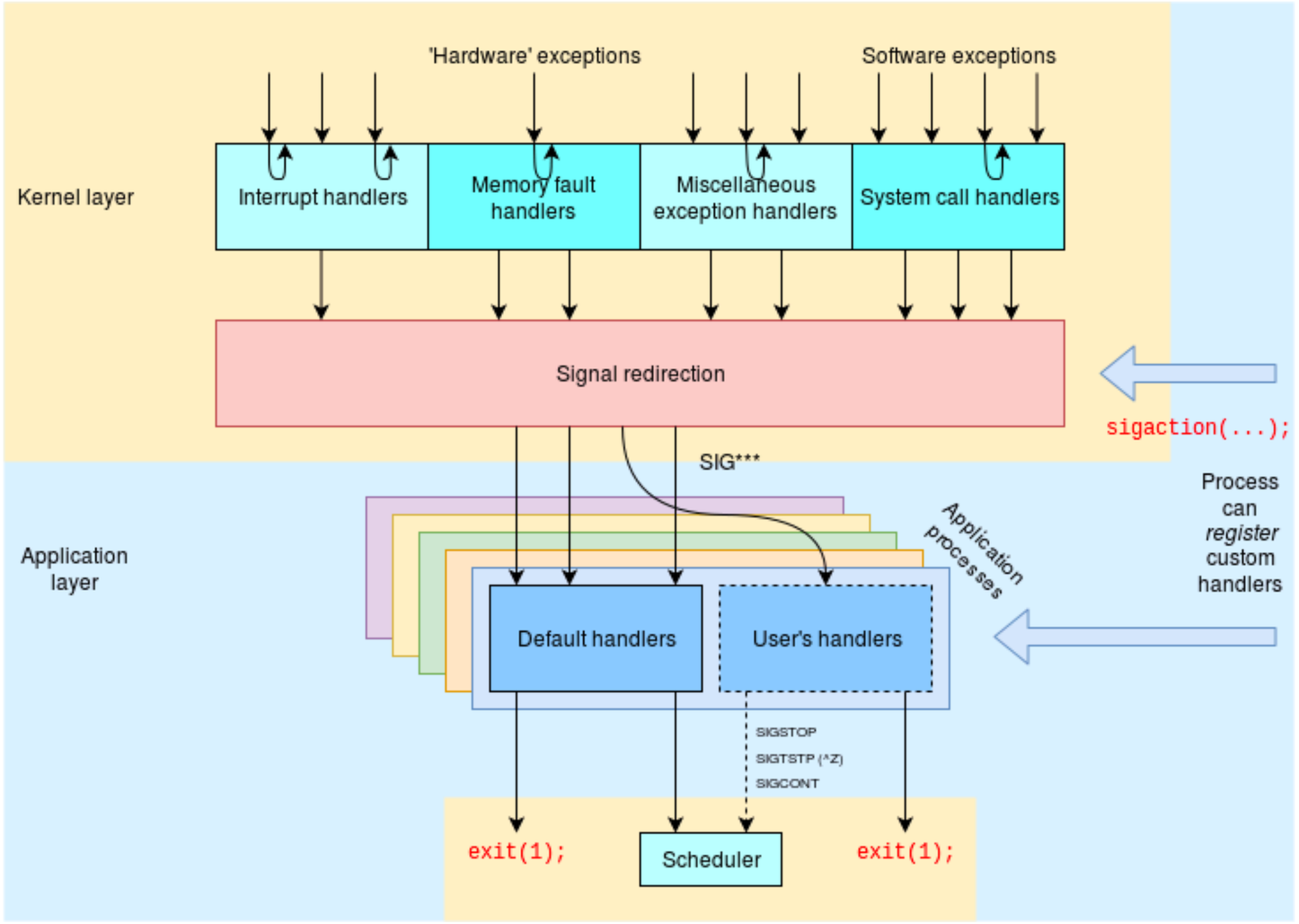


User Space

Kernel Space

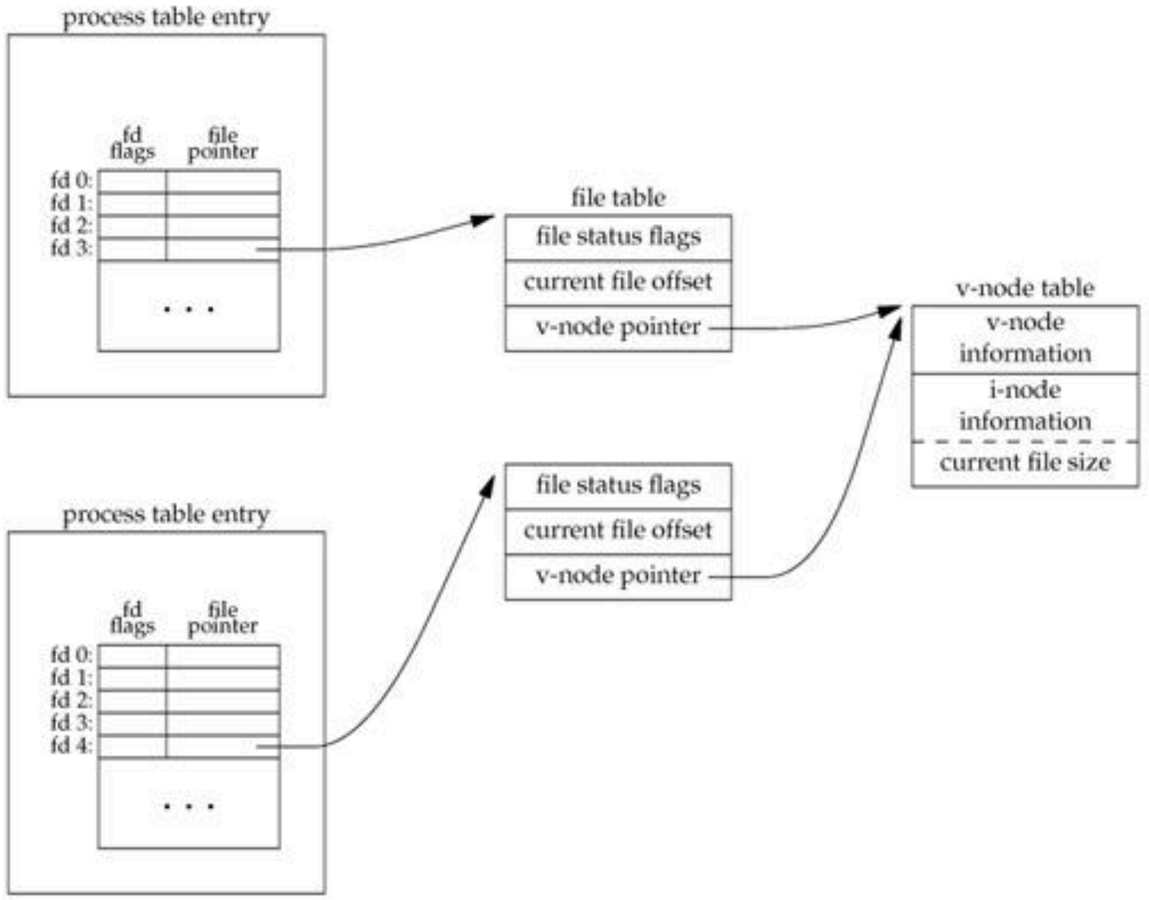


Сигналы, синхронные и асинхронные




```
>kill -l
 1) SIGHUP       2) SIGINT       3) SIGQUIT      4) SIGILL
 5) SIGTRAP      6) SIGABRT      7) SIGEMT       8) SIGFPE
 9) SIGKILL     10) SIGBUS      11) SIGSEGV     12) SIGSYS
13) SIGPIPE     14) SIGALRM     15) SIGTERM     16) SIGURG
17) SIGSTOP     18) SIGTSTP     19) SIGCONT     20) SIGCHLD
21) SIGTTIN     22) SIGTTOU     23) SIGIO       24) SIGXCPU
25) SIGXFSZ     26) SIGVTALRM   27) SIGPROF     28) SIGWINCH
29) SIGINFO     30) SIGUSR1     31) SIGUSR2
```

Ресурсы, файлы

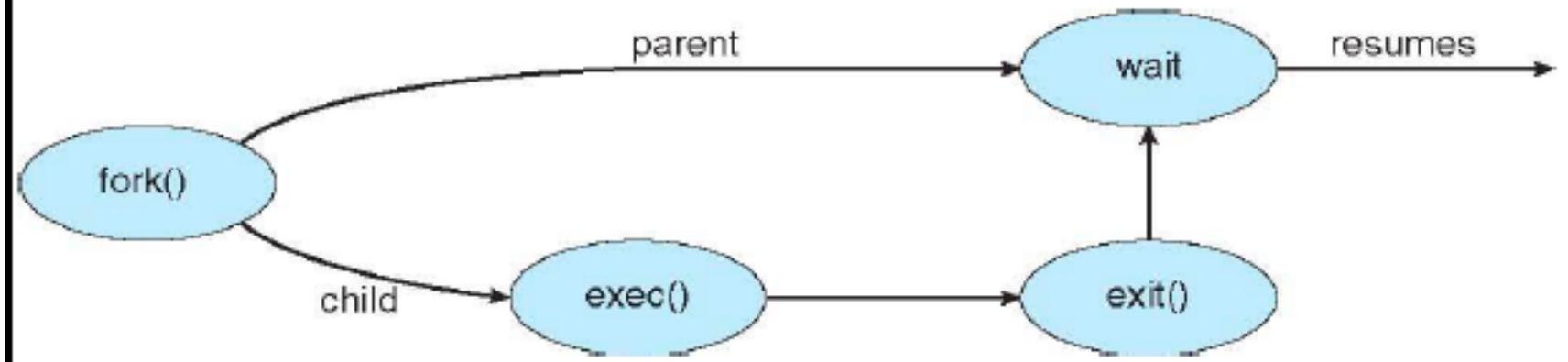


Освобождение ресурсов

- Любое завершение процесса (exit syscall, fatal sync or async signal)
- Вся работа по освобождению ресурсов в ядре
- Освобождаются
  - память
  - файловые дескрипторы (файлы, советы)
  - семафоры
- Не освобождаются:
  - разделяемая память (shmget(2))
  - код выхода (wait(2)), процессы-зомби

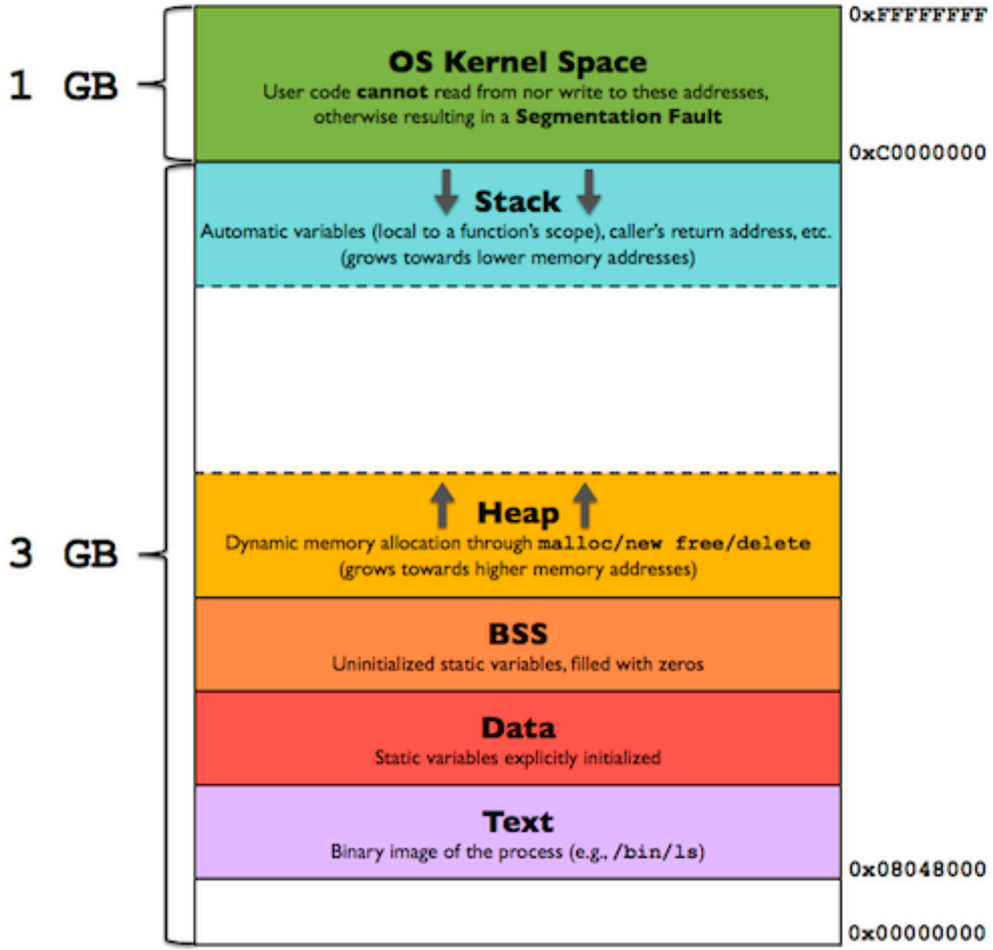
fork(2) и exec(2)

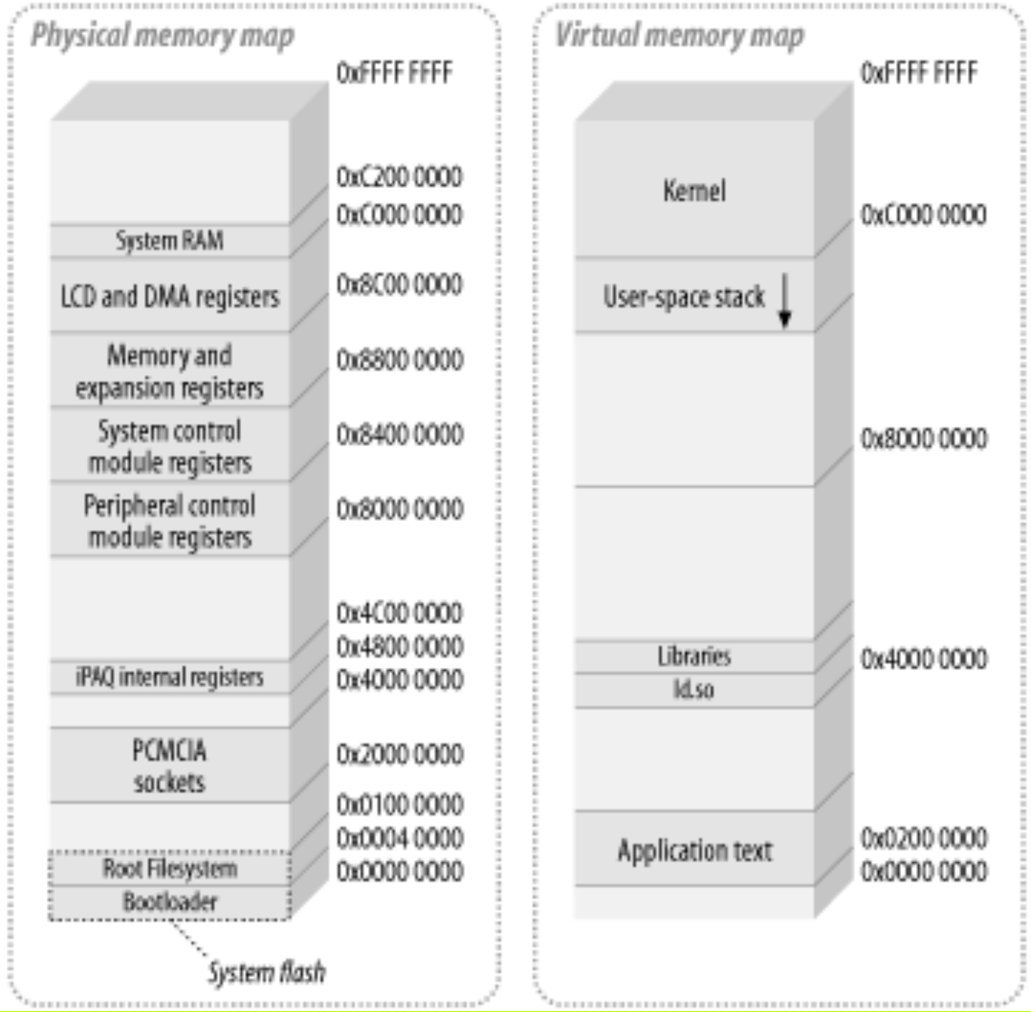
- Address space
  - Child duplicate of parent
  - Child has a program loaded into it
- UNIX examples
  - `fork()` system call creates new process
  - `exec()` system call used after a `fork()` to replace the process' memory space with a new program



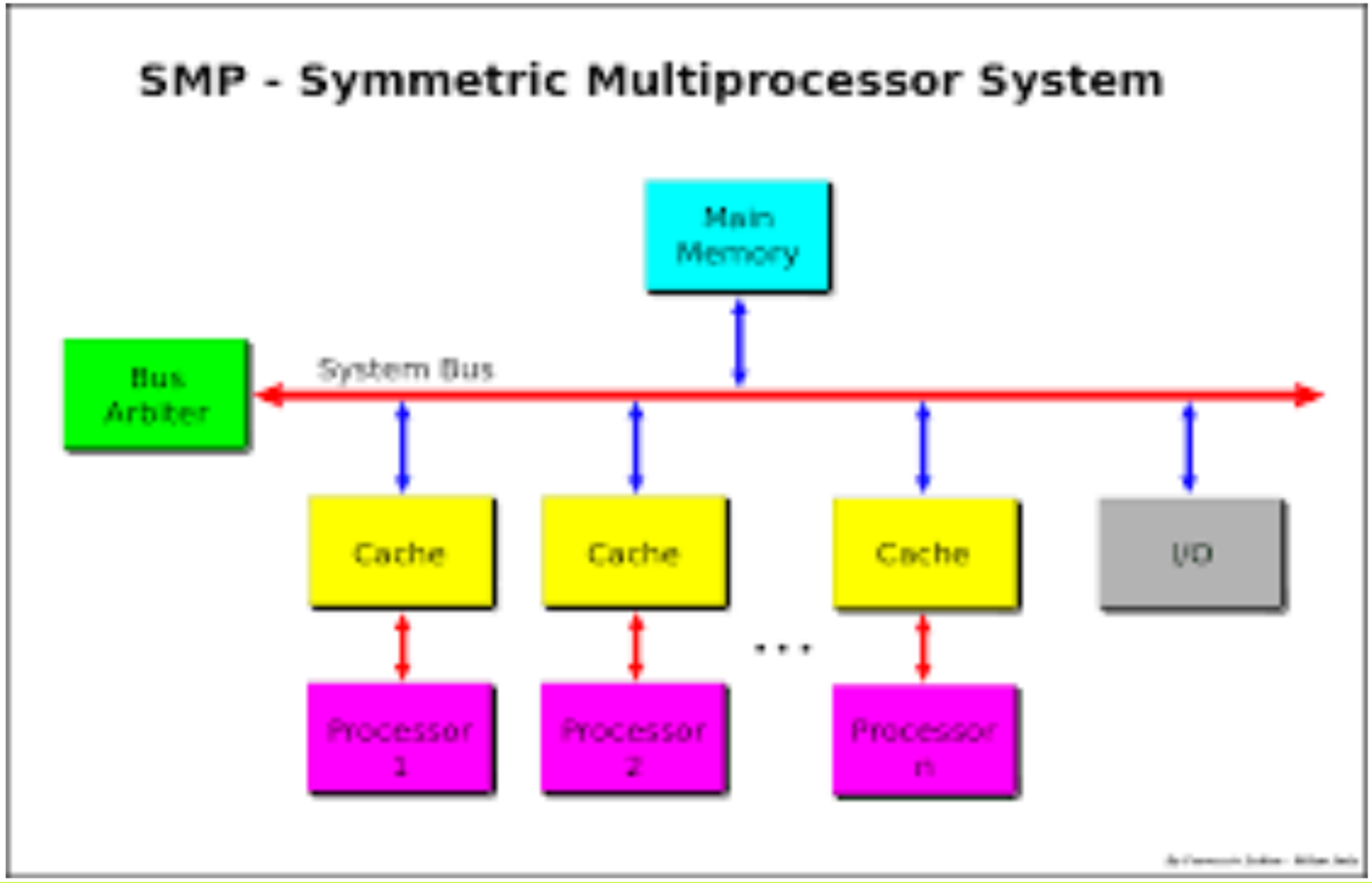
posix_spawn(2)

Организация памяти процесса

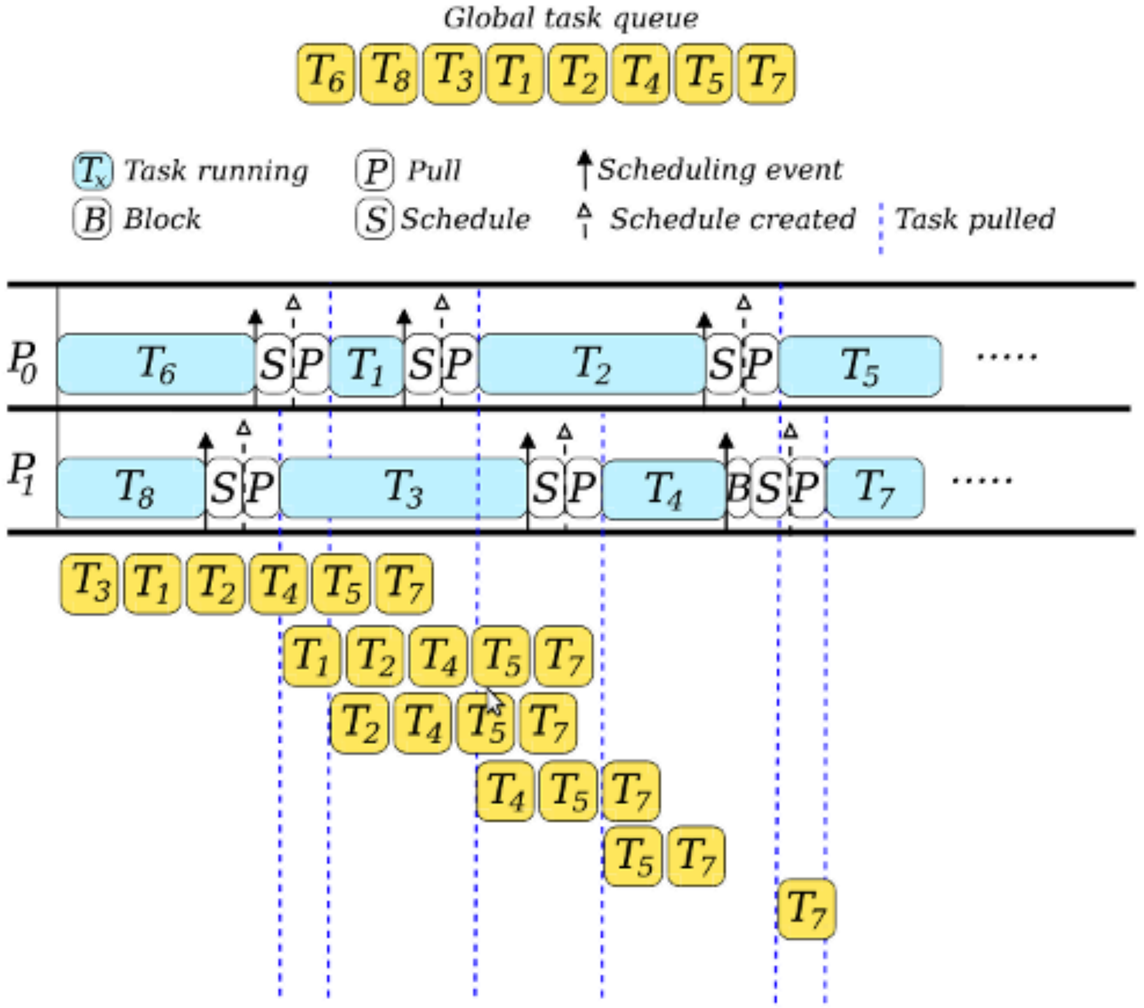




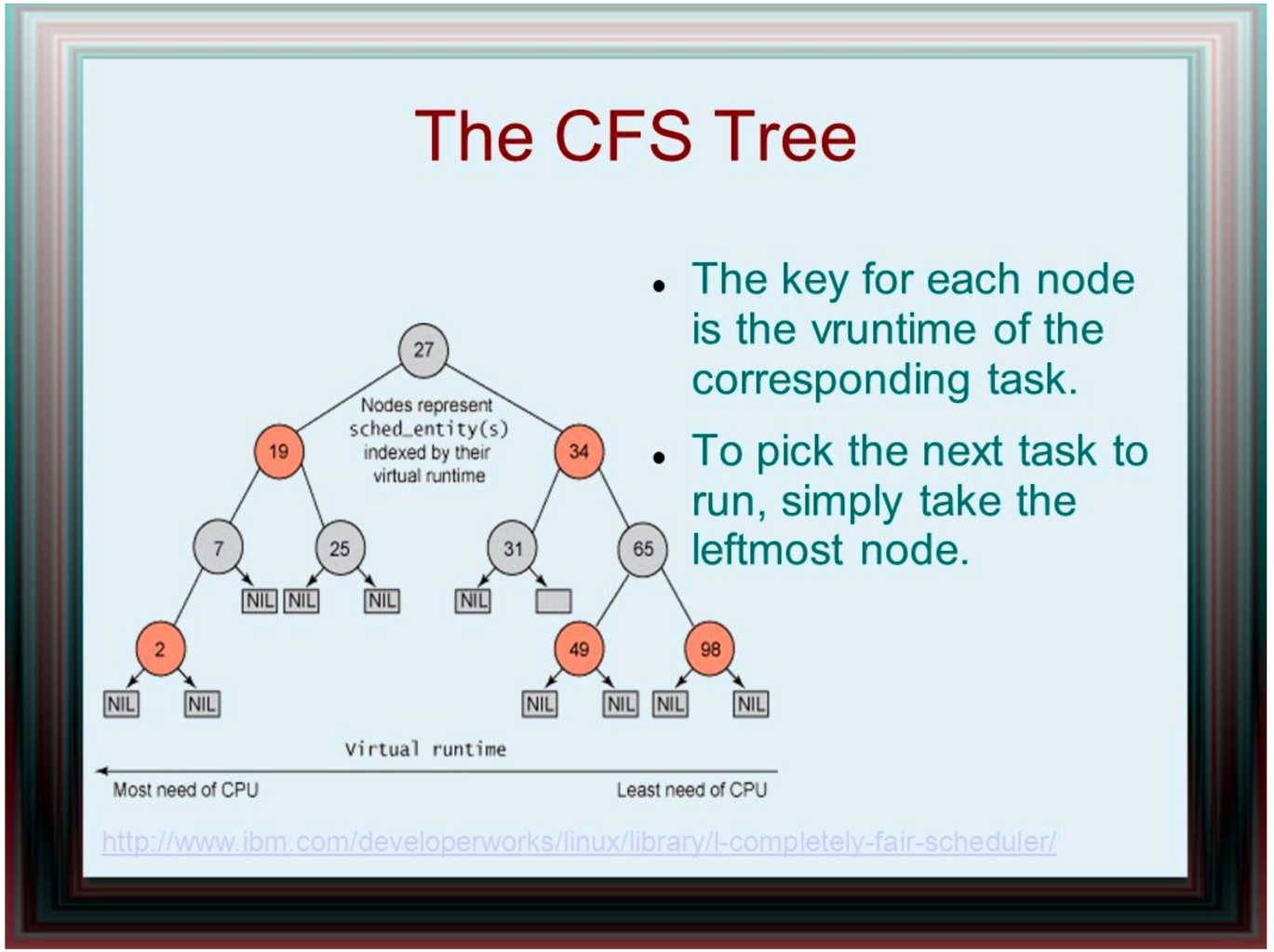
SMP, NUMA


SMP - Symmetric Multiprocessor System
Main Memory
Bus Arbiter
System Bus
Cache
Cache
Cache
I/O
Processor 1
Processor 2
Processor n

Планировщик



Алгоритмы планировщика *



* HTTPS://OPENSOURCE.COM/ARTICLE/19/2/FAIR-SCHEDULING-LINUX

Настройка планировщика

- Приоритеты процессов/потоков
- Группы планирования
- Вычислительно-интенсивные и IO-ориентированные процессы
- Process affinity и кеши
- Процессы реального времени и sched_setscheduler(2)
- Потенциальные проблемы: starvation, priority inversion

«To the optimist, the glass is half full. To the pessimist, the glass is half empty. To the engineer, the glass is twice as big as it needs to be.»

Anonymous