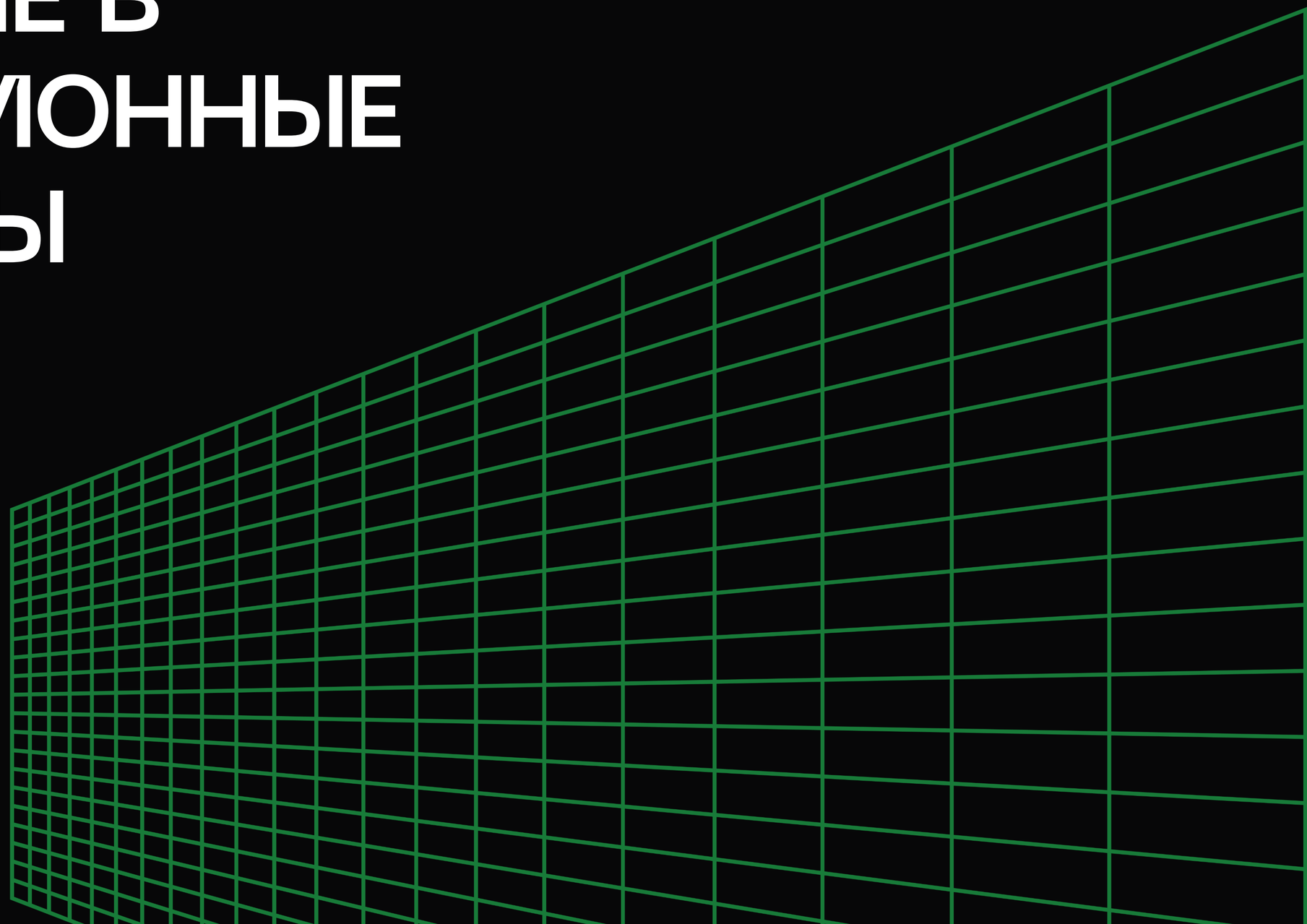
ВВЕДЕНИЕ В ОПЕРАЦИОННЫЕ СИСТЕМЫ



Balun.Courses

Balun.Courses

ЗАПИСЬ

Balun.Courses

ПРАВИЛА ЗАНЯТИЯ

1. Вопросы можно и нужно задавать в любое время, если что-то непонятно
2. Лучше всего задавать вопросы голосом

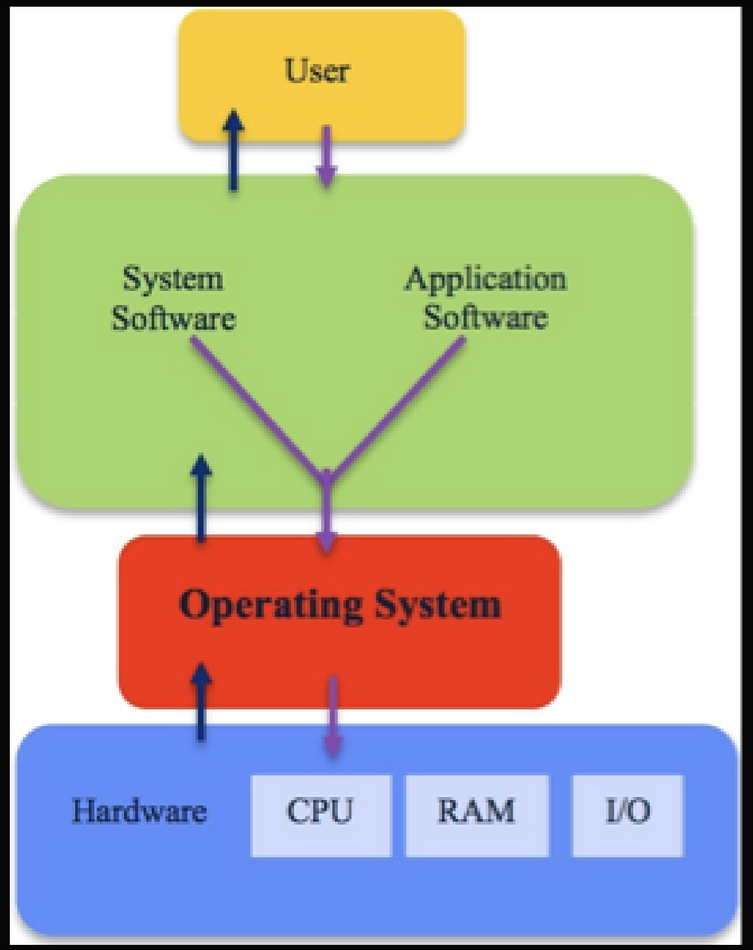
Balun.Courses

ПЛАН ЛЕКЦИИ

1. Мотивация изучения операционных систем
2. Представление операционных систем в Go
3. Архитектура ядра операционной системы
4. SMP, ASMP, сравнение
5. Абстракции операционной системы

Balun.Courses

МОТИВАЦИЯ

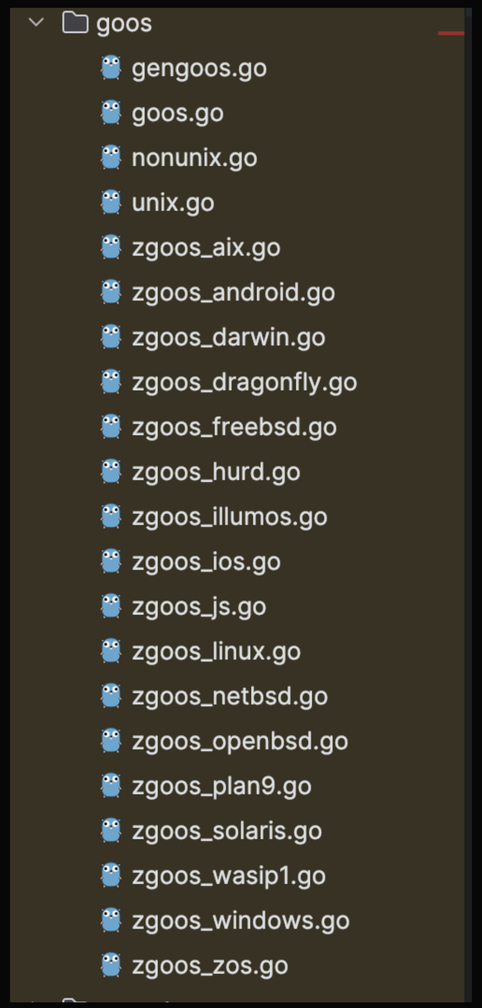


Операционная система управляет аппаратной частью компьютера, программным обеспечением и предоставляет интерфейсы для различных программ

BSD (Free BSD, Open BSD), Unix, Linux, iOS (macOS), Zephyr, etc.

Balun.Courses

ПРЕДСТАВЛЕНИЕ В GO




```go
func main() {
    fmt.Println(runtime.GOOS)
    fmt.Println(runtime.GOARCH)
}
```

```
/Users/igorwalther/Library/Caches/JetBrains/GoLand2024.1/tmp/GoLand/___go_build_fork
darwin
arm64
```

(GOARCH=arm64 GOOS=darwin go build main.go)

Balun.Courses

ПРЕДСТАВЛЕНИЕ В GO

GOOS: aix, android, darwin, dragonfly, freebsd, illumos,
ios, js, linux, netbsd, openbsd, wasip1, plan9, solaris, wasip1,
windows

GOARCH: (386, amd64, arm, arm64, loong64, mips, mips64,
mips64le, mipsle, ppc64, ppc64le, riscv64, s390x, wasm)

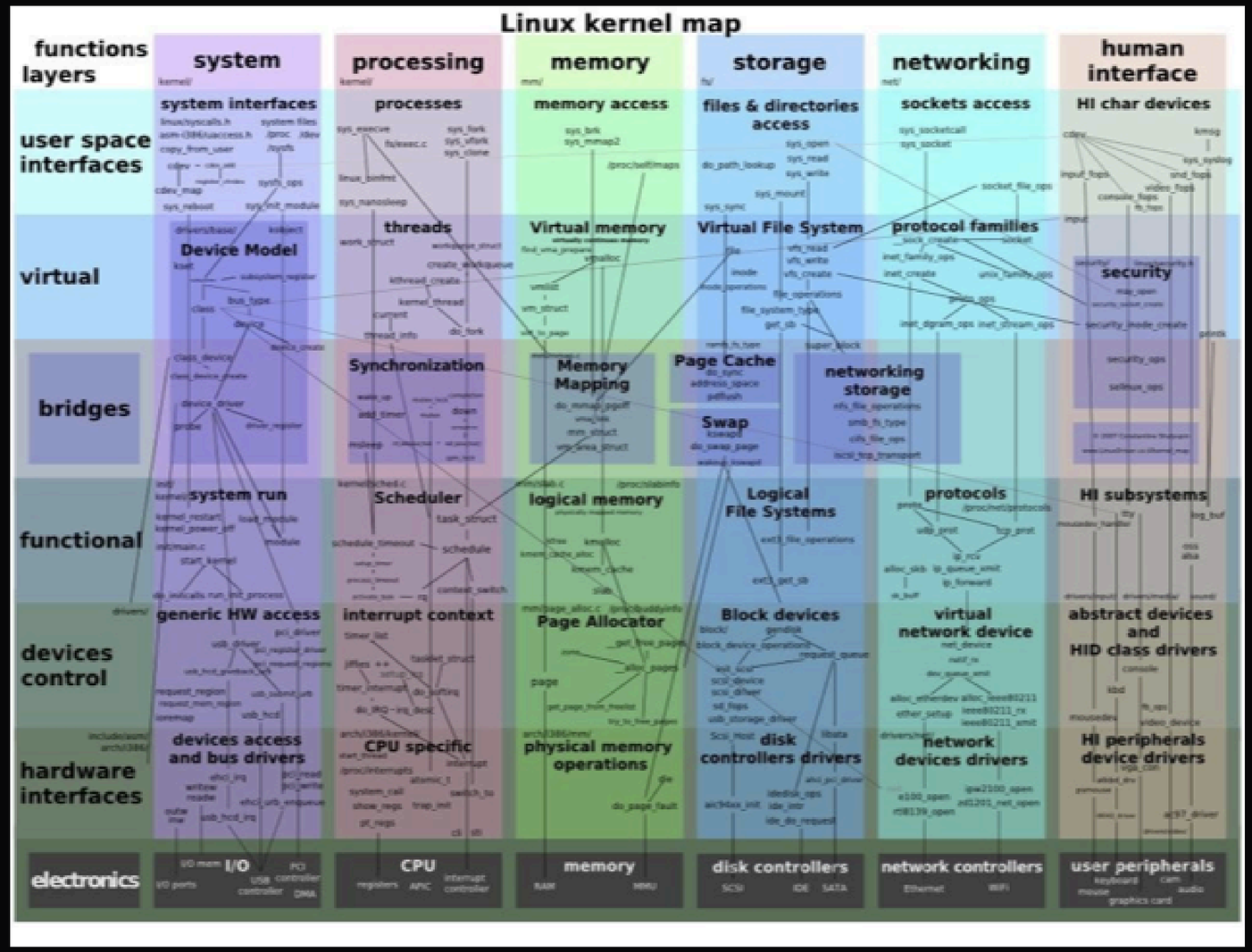
```
go tool dist list
```

Balun.Courses

СЛОЖНОСТЬ ЯДРА OS

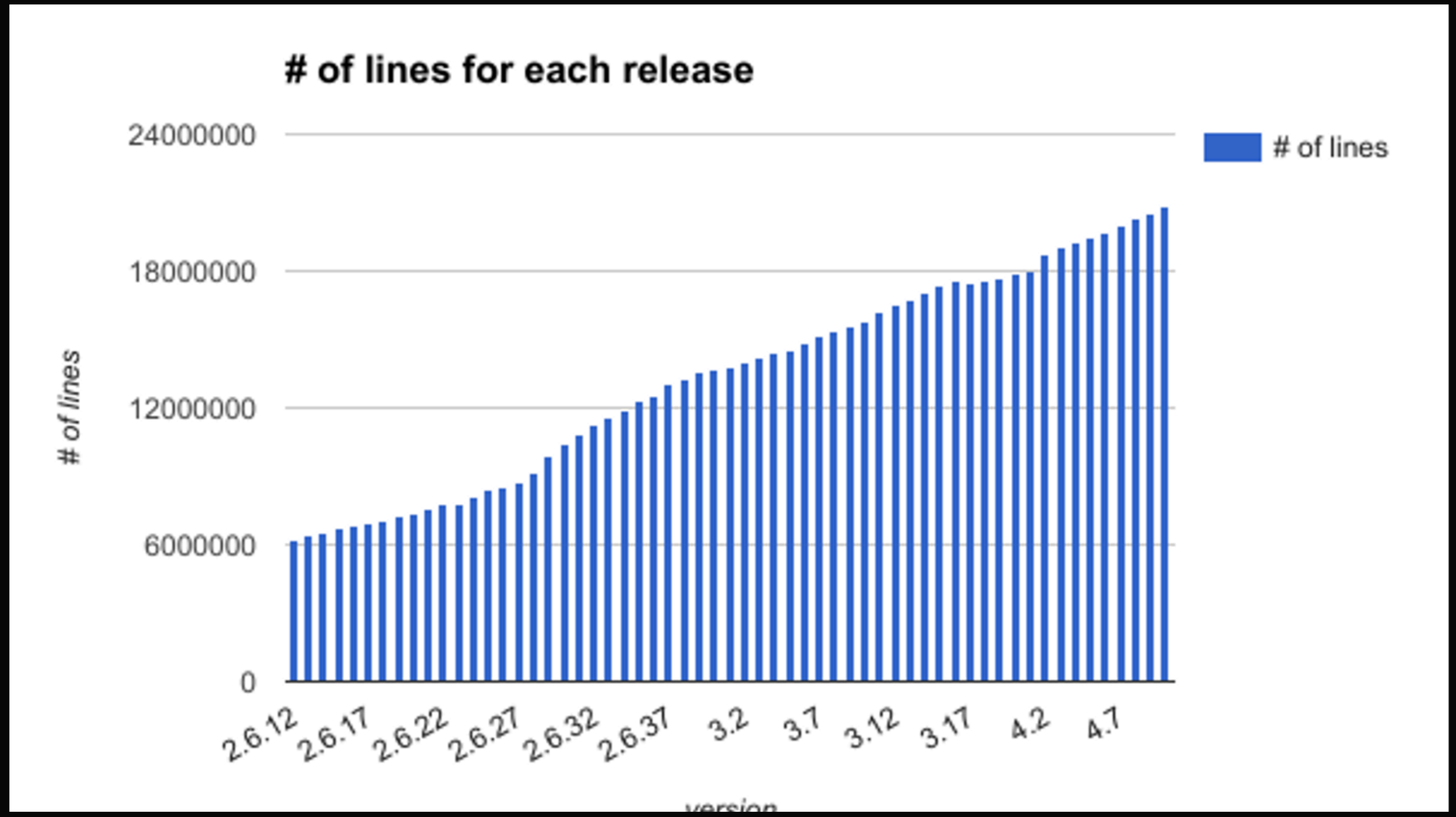
Ядро является самой сложной частью операционной системы

В чем отличие между ядром и операционной системой?



Balun.Courses

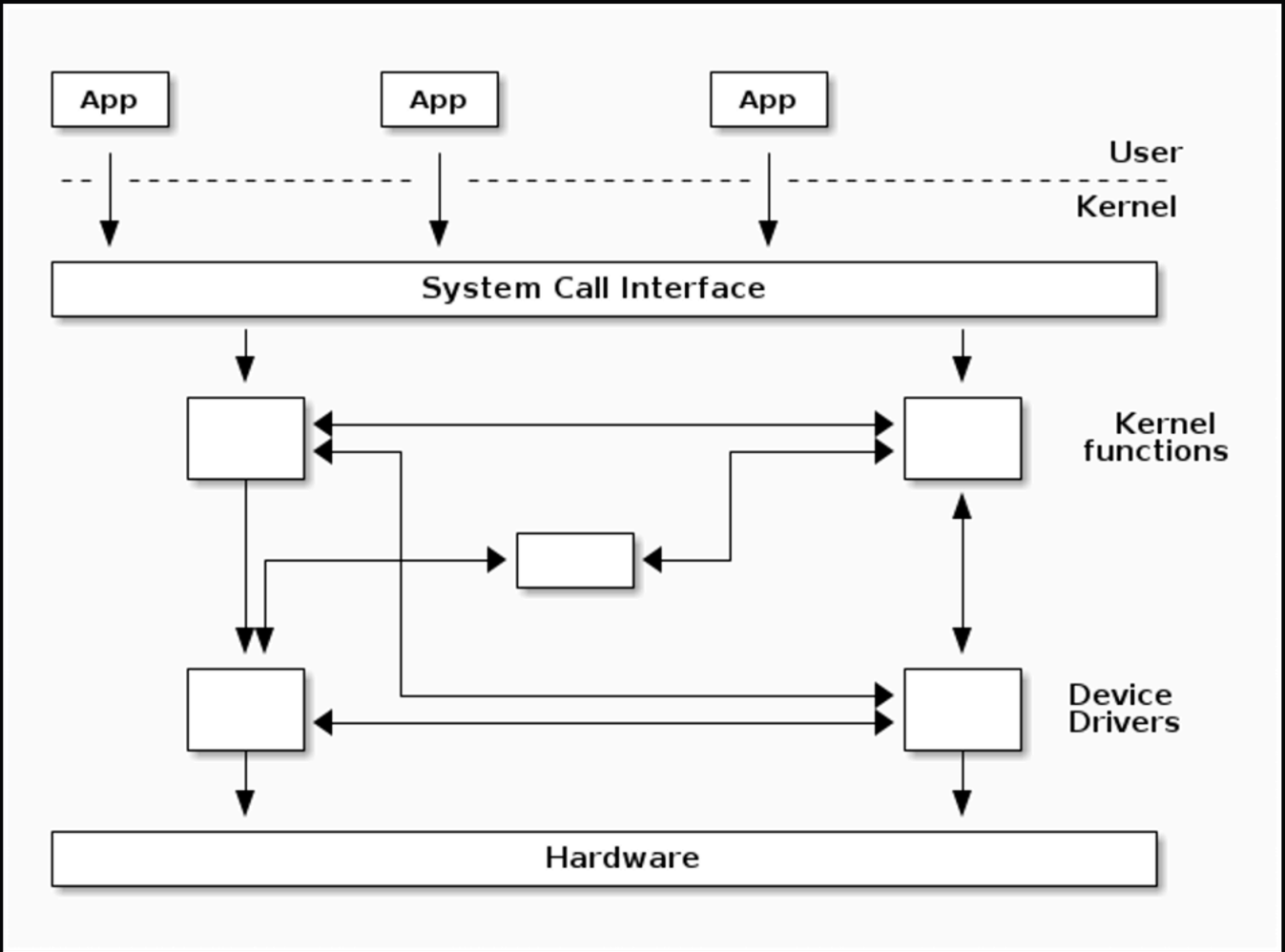
СЛОЖНОСТЬ ЯДРА OS



Количество строк ядра Linux

Balun.Courses

АРХИТЕКТУРА ЯДРА

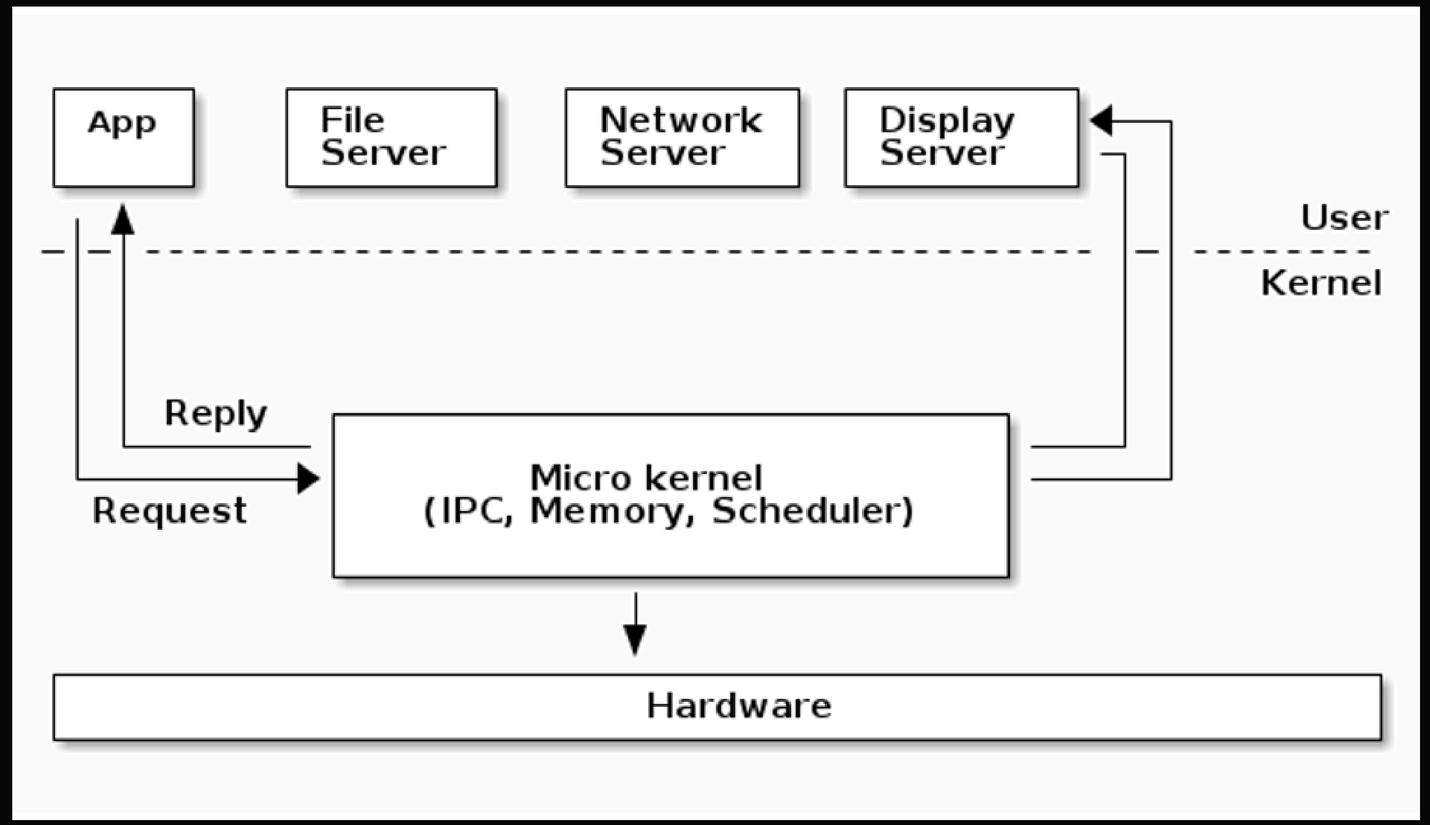


В монолитной архитектуре:

1. Нету контроля доступа между разными компонентами ядра
2. Логическое разделение между подсистемами посредством API
3. Есть поддержка модулярности в compile time
4. Есть поддержка загружаемых модулей во время исполнения
5. Строгие интерфейсы с маленьким количеством накладных расходов

Balun.Courses

АРХИТЕКТУРА ЯДРА



В микросервисной архитектуре:

1. Большинство подсистем изолированы друг от друга, в частности как пользовательские процессы
2. Изолированная память между сервисами в обмен на производительность

Balun.Courses

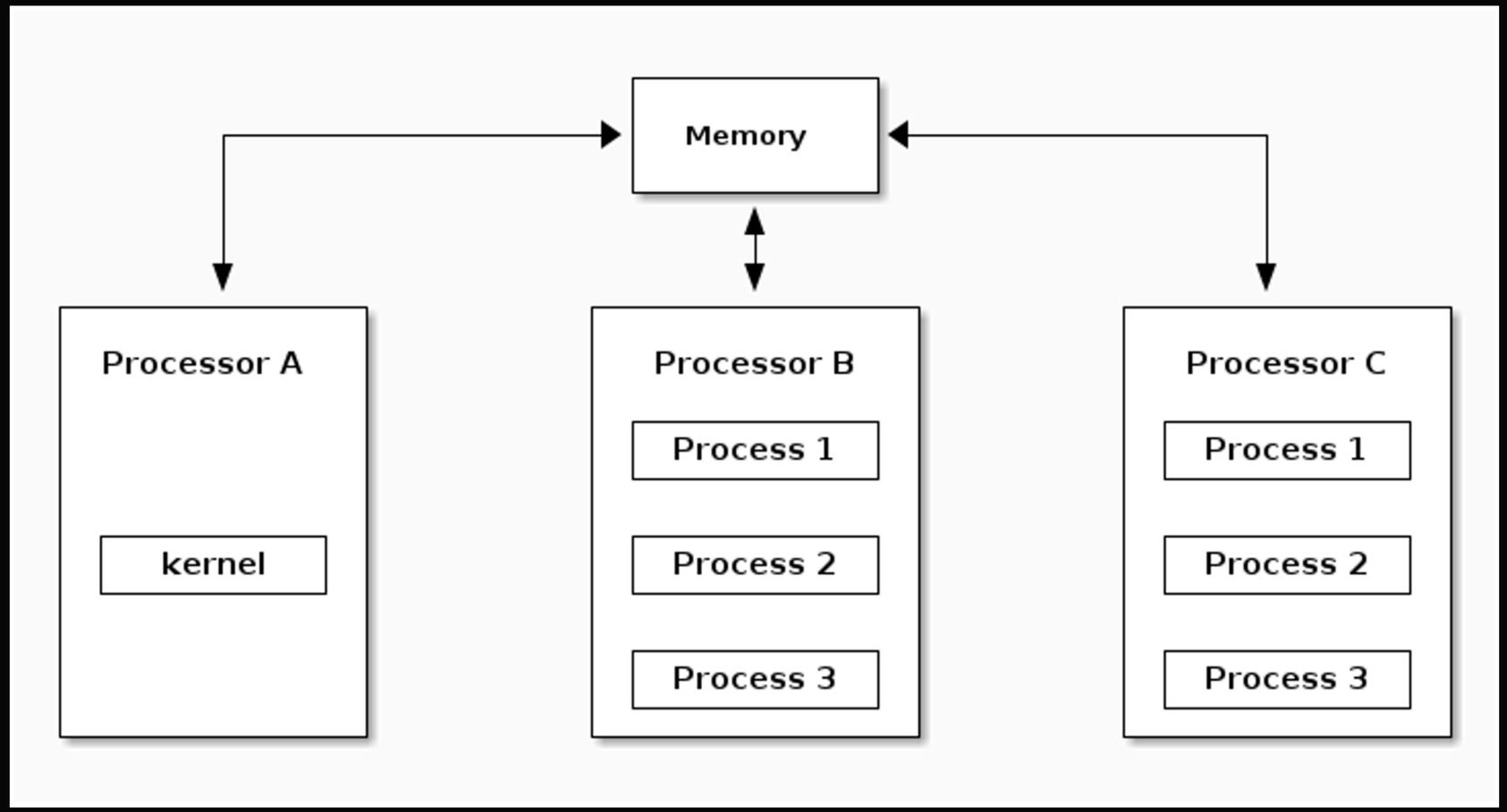
СРАВНЕНИЕ АРХИТЕКТУР

1. Монолитные ядра быстрее
2. Микросервисные ядра более надежные, но медленные
3. Есть смешанный подход, который использует Windows

LWN articleon Monoliticvs.Micro (2007)

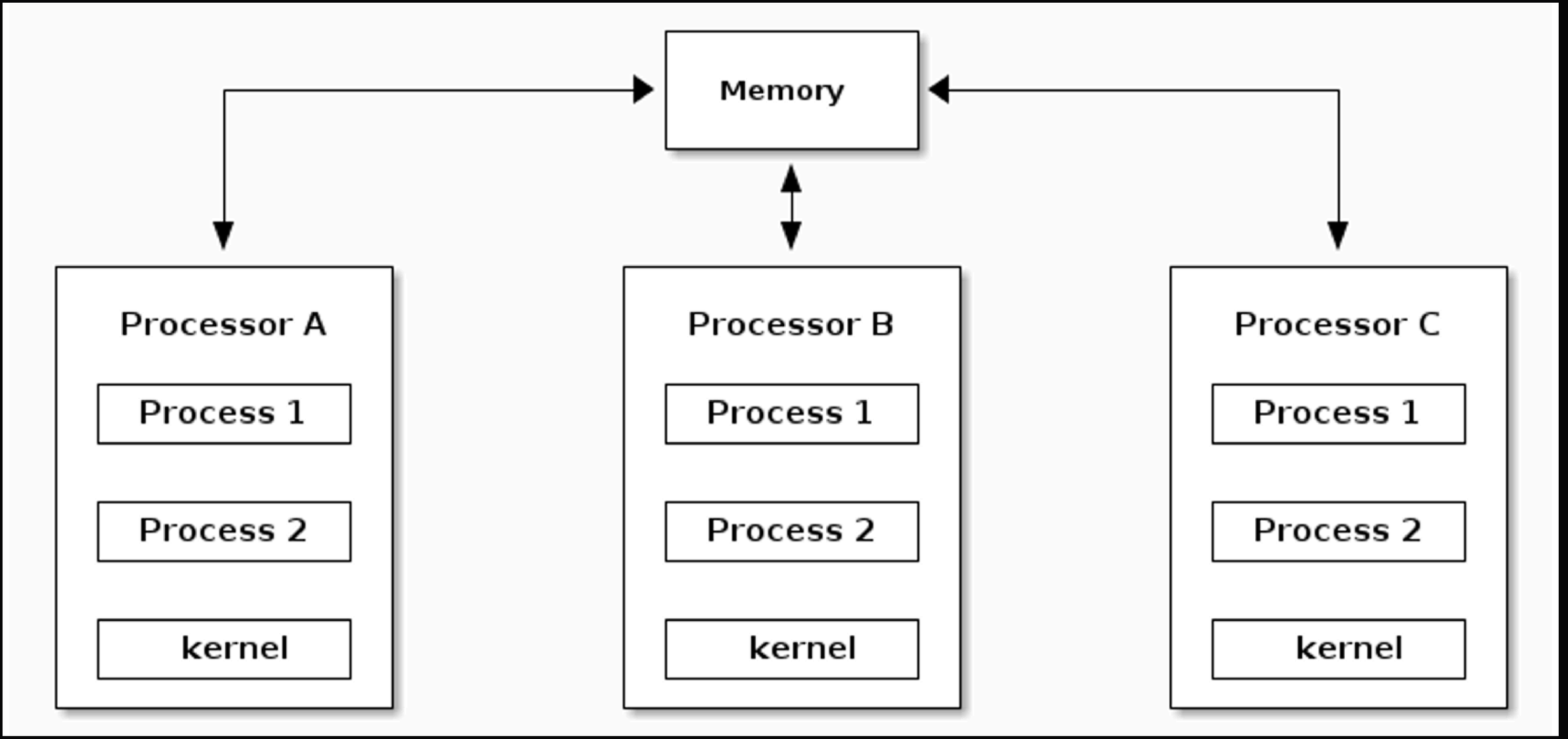
Balun.Courses

ASYMMETRIC MULTIPROCESSING (ASMP)



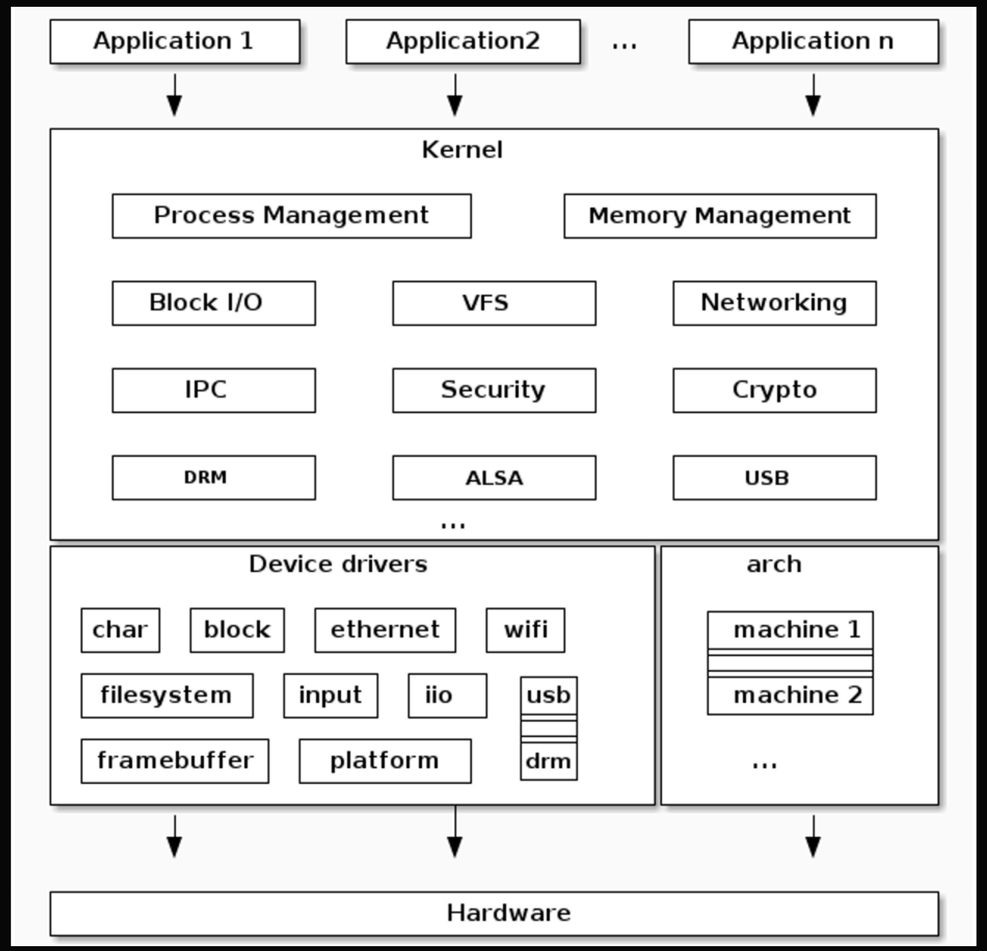
Balun.Courses

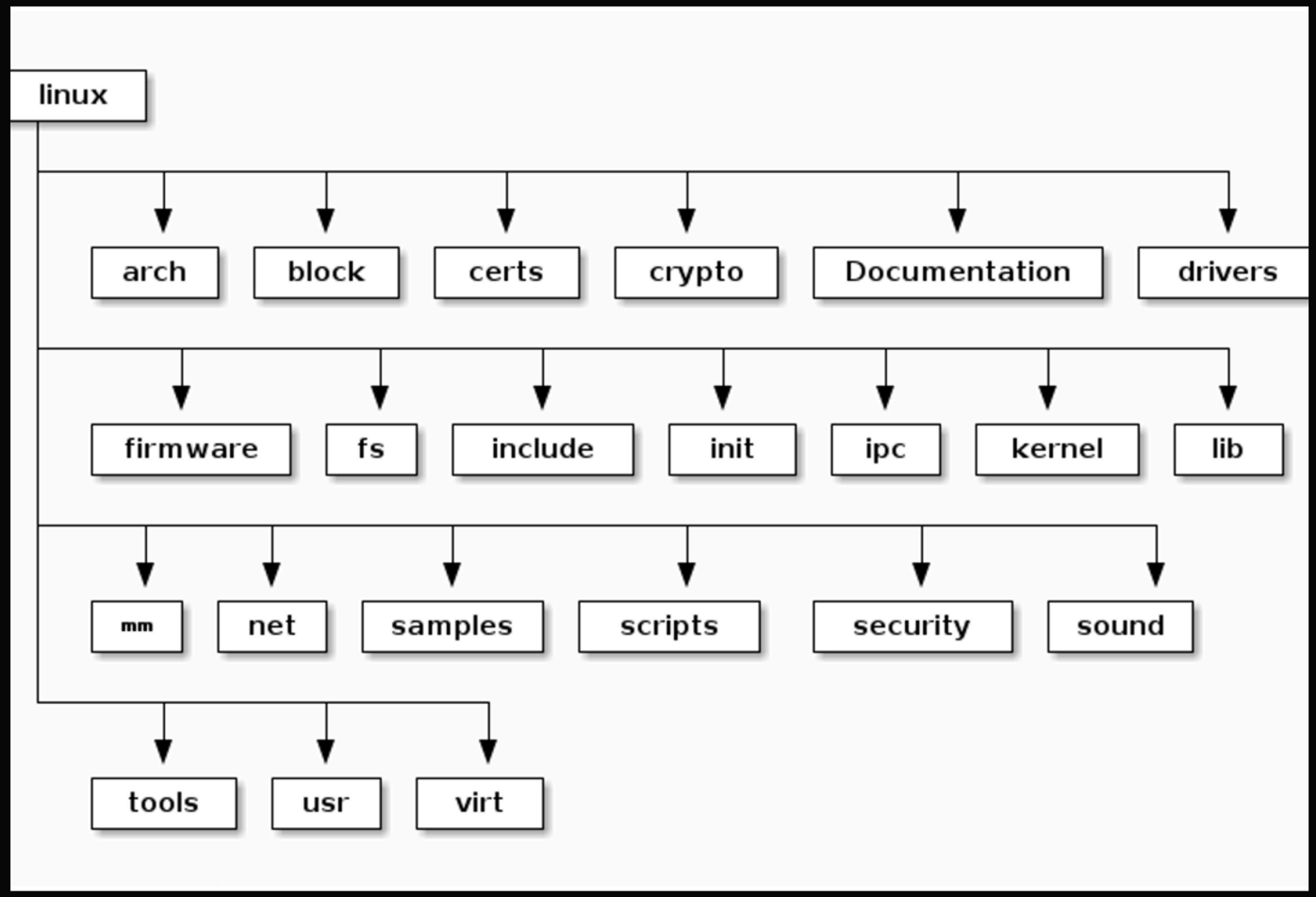
SYMMETRIC MULTI-PROCESSING



Balun.Courses

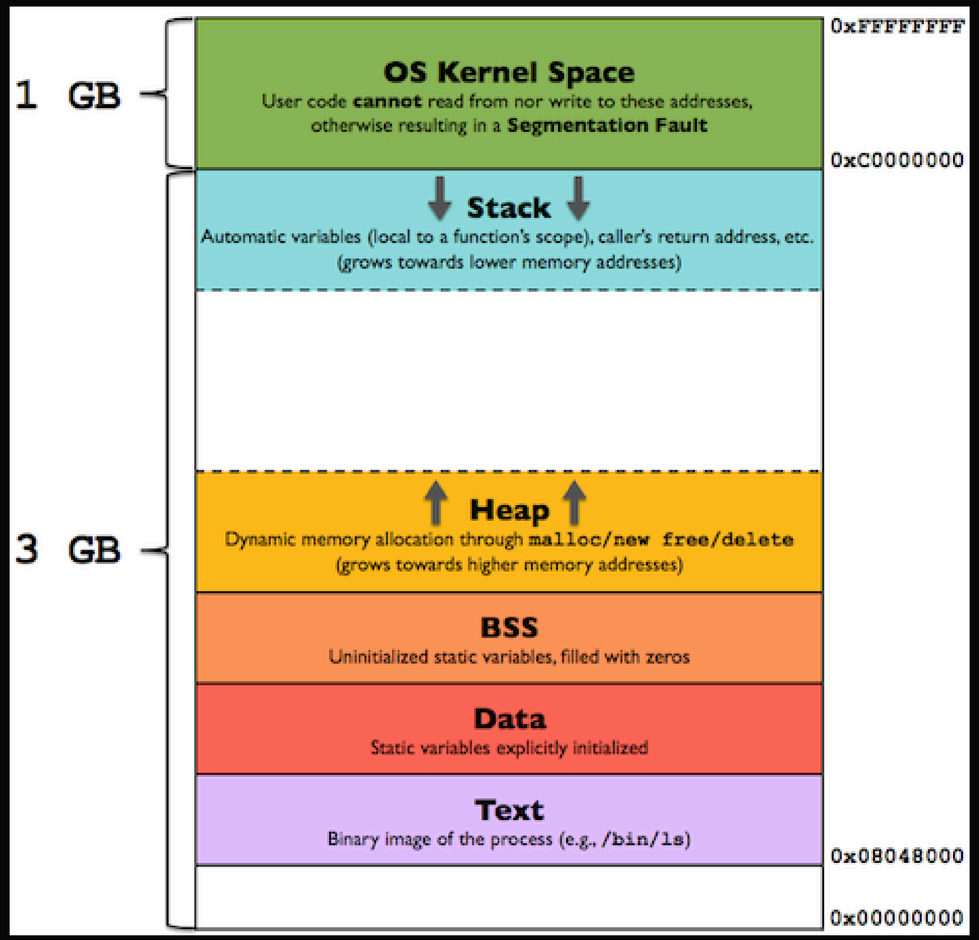
АРХИТЕКТУРА ЯДРА LINUX





Balun.Courses

АБСТРАКЦИИ ОПЕРАЦИОННОЙ СИСТЕМЫ



Процесс это абстракция операционной системы - программа для исполнения, включает в себя:

1. Адресное пространство
2. Один или несколько потоков
3. Дескрипторы
4. Семафоры
5. Таймеры
6. Обработчики сигналов
7. Прочие ресурсы/метаинформация

Balun.Courses

АБСТРАКЦИИ ОПЕРАЦИОННОЙ СИСТЕМЫ

```go
func main() {
    foo()
}

func foo() {  2 usages
    bar()
}

func bar() {  1 usage
    stackDump := debug.Stack()
    fmt.Println(string(stackDump), len(stackDump))
    fmt.Println( a...: "")
}
```

```
runtime/debug.Stack()
        /Users/igorwalther/.ya/tools/v4/6067021677/src/runtime/debug/stack.go:24 +0x64
main.bar()
        /Users/igorwalther/GolandProjects/performance_optimizations_go/lection_examples/lection_4/fork/main.go:17 +0x1c
main.foo(...)
        /Users/igorwalther/GolandProjects/performance_optimizations_go/lection_examples/lection_4/fork/main.go:13
main.main()
```

Balun.Courses

АБСТРАКЦИИ ОПЕРАЦИОННОЙ СИСТЕМЫ

```go
// On Unix systems, FindProcess always succeeds and returns a Process
// for the given pid, regardless of whether the process exists. To test whether
// the process actually exists, see whether p.Signal(syscall.Signal(0)) reports
// an error.
func FindProcess(pid int) (*Process, error) {
    return findProcess(pid)
}
```

```go
// StartProcess is a low-level interface. The os/exec package provides
// higher-level interfaces.
//
// If there is an error, it will be of type *PathError.
func StartProcess(name string, argv []string, attr *ProcAttr) (*Process, error) {
    testlog.Open(name)
    return startProcess(name, argv, attr)
}
```

Как рантайм Go понимает, где расположен исполняемый файл?

Balun.Courses

АБСТРАКЦИИ ОПЕРАЦИОННОЙ СИСТЕМЫ

```go
func main() {
    p, err := os.Executable()

    if err != nil {
        log.Fatal(err)
    }

    fmt.Println(p)
    fmt.Println(os.Args[0])
}
```

```
/Users/igorwalther/Library/Caches/JetBrains/GoLand2024.1/tmp/GoLand/___go_build_fork
/Users/igorwalther/Library/Caches/JetBrains/GoLand2024.1/tmp/GoLand/___go_build_fork
```

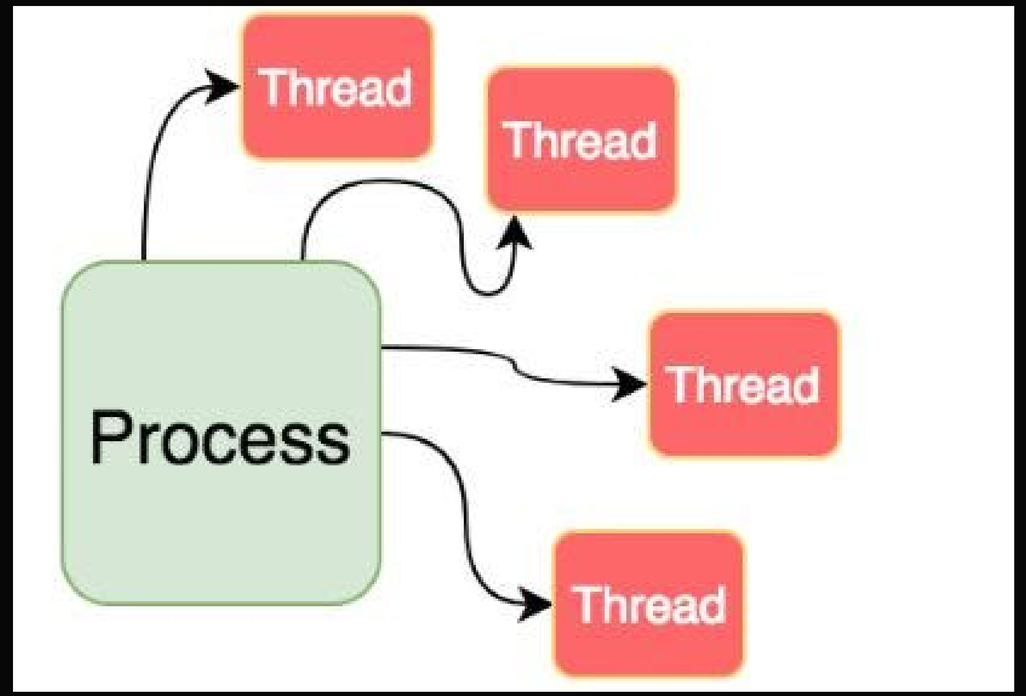
Используется для различных утилит, чтобы найти исполняемый файл, например, pprof

Balun.Courses

АБСТРАКЦИИ ОПЕРАЦИОННОЙ СИСТЕМЫ

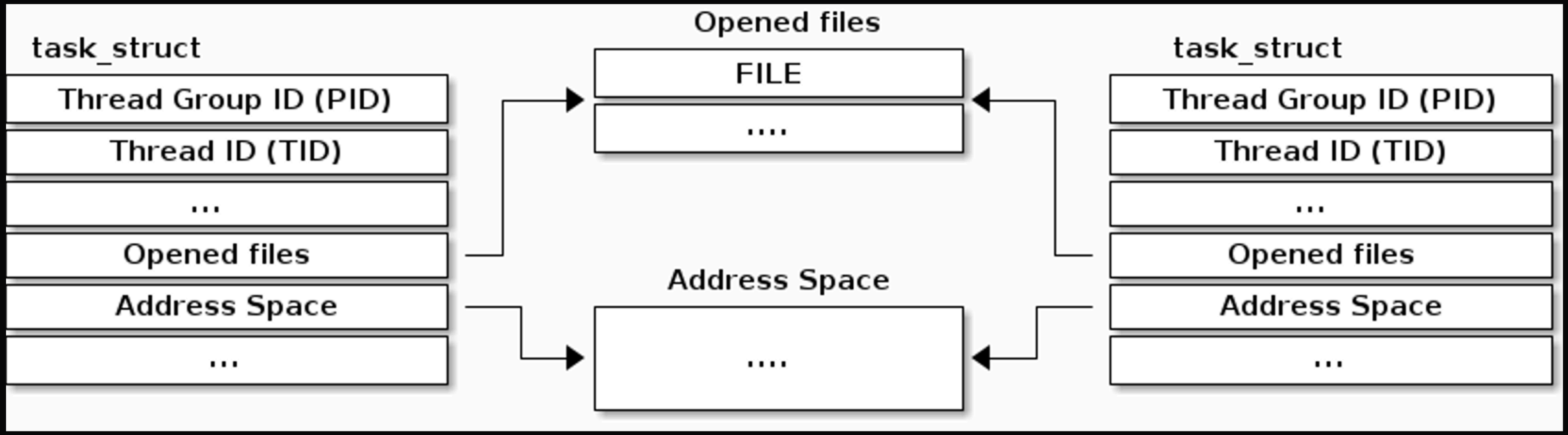
Поток это абстракция операционной системы при исполнении программы.

1. Процесс может выполнять инструкции конкуретно, создавая один или несколько потоков
2. Потоки в рамках одного процесса имеют общие ресурсы, в том числе память.
3. У каждого потока есть свой собственный стек и набор регистров



Balun.Courses

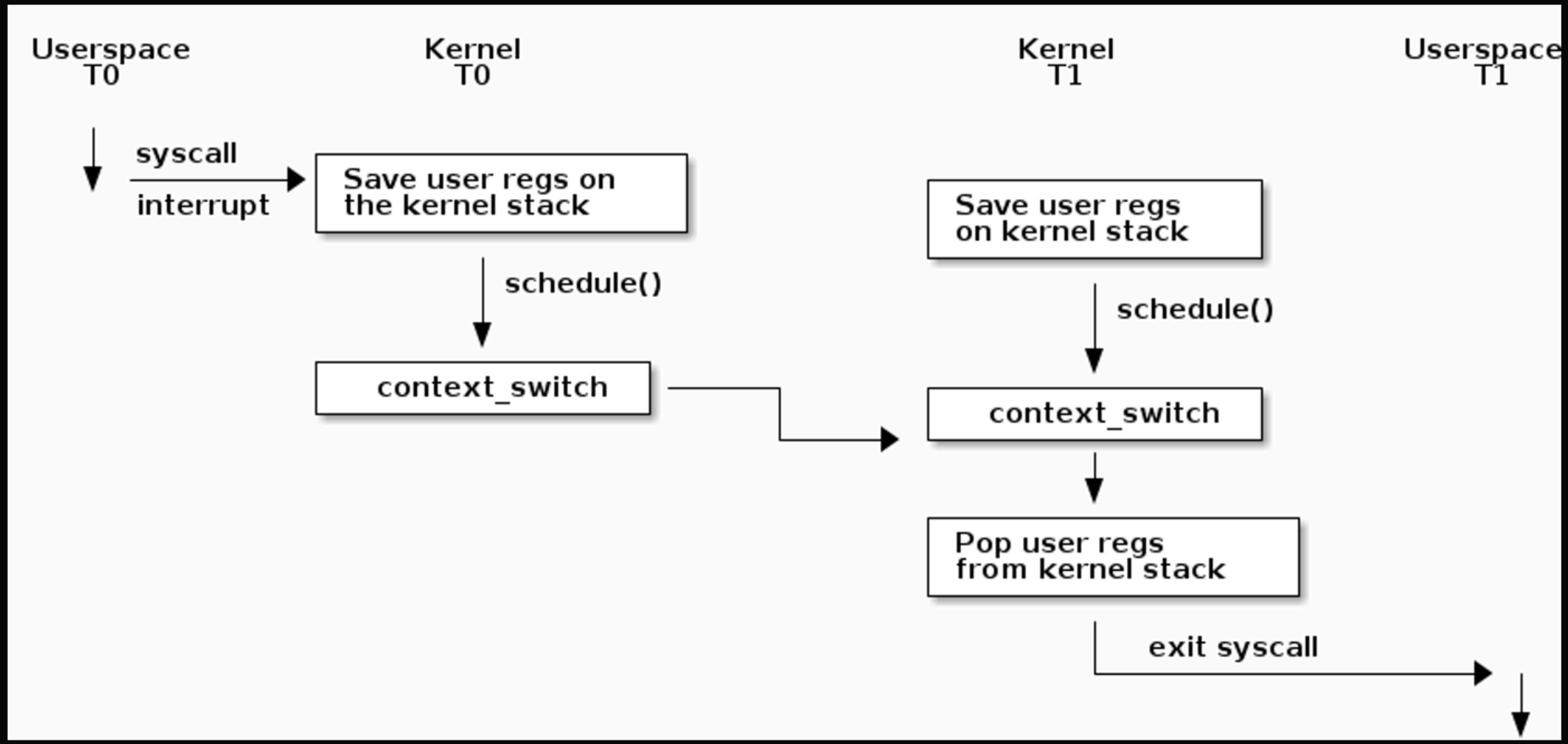
АБСТРАКЦИИ ОПЕРАЦИОННОЙ СИСТЕМЫ



Реализация потоков в Linux

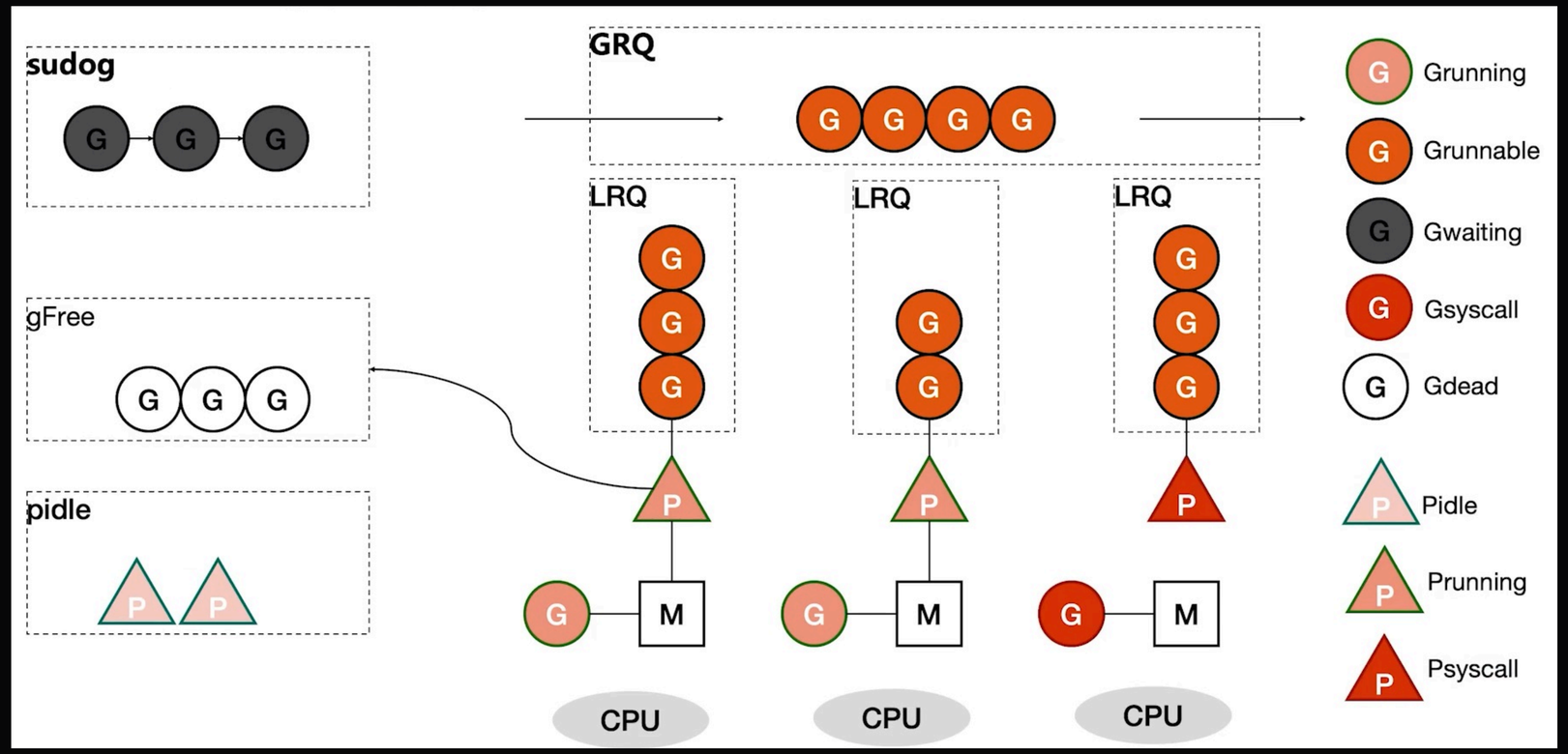
Balun.Courses

АБСТРАКЦИИ ОПЕРАЦИОННОЙ СИСТЕМЫ



Balun.Courses

АБСТРАКЦИИ ОПЕРАЦИОННОЙ СИСТЕМЫ В GO



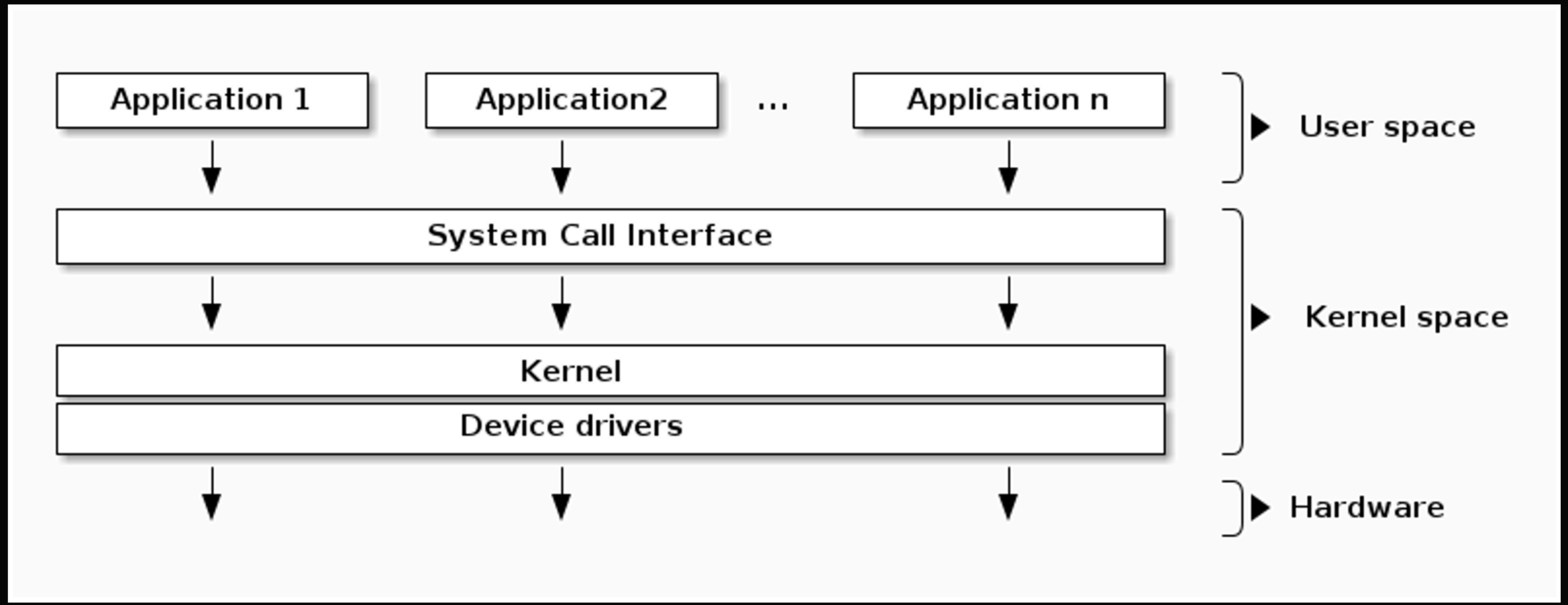

```
➜  ~ sysctl hw.logicalcpu
hw.logicalcpu: 12
```

```go
func main() {
    fmt.Println(runtime.NumCPU()) // 12
}
```

Почему логических ядер может быть больше, чем физических?

Balun.Courses

ВЗАИМОДЕЙСТВИЕ С OS



ИТОГИ И ВЫВОДЫ

мой Telegram

Linkedin