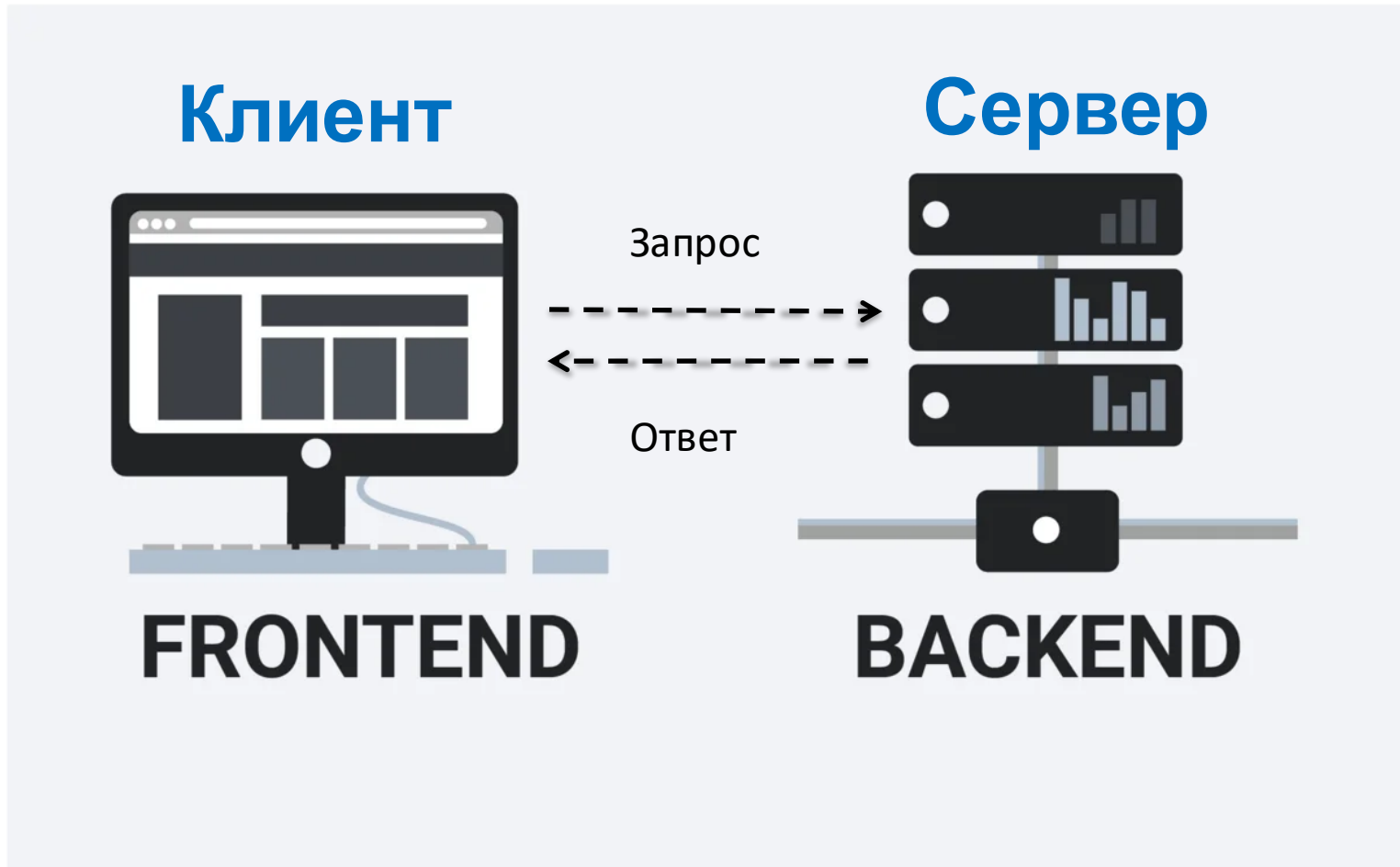
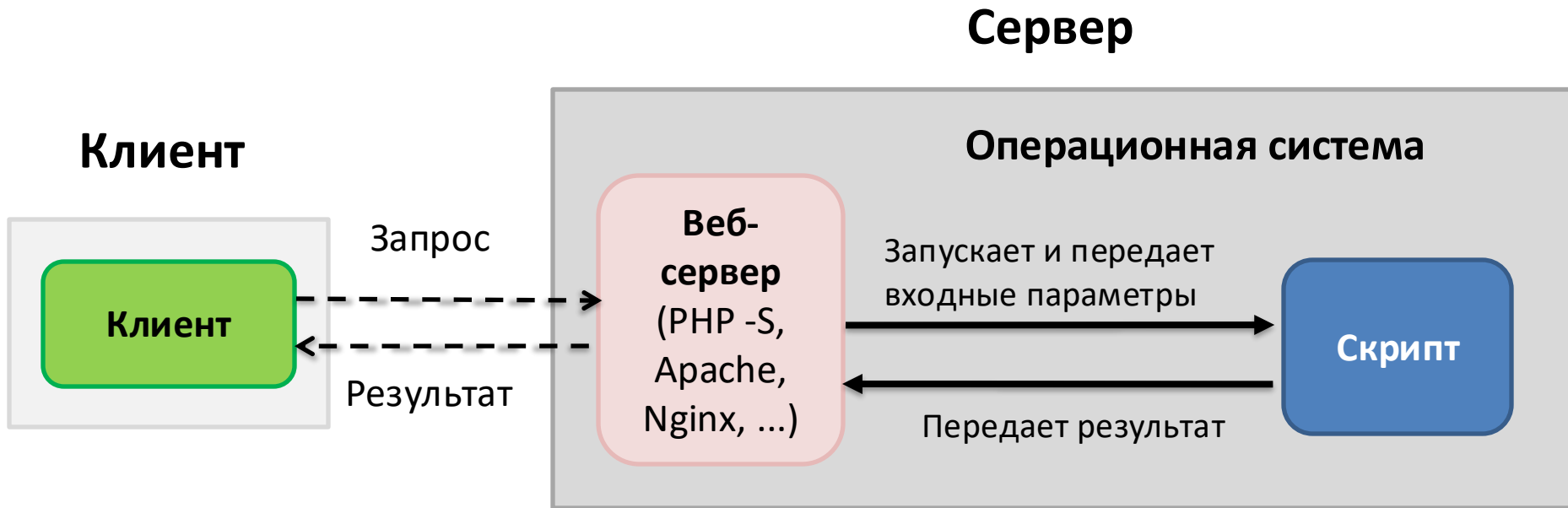


## **2. Серверная операционная система. Консольные приложения на PHP**

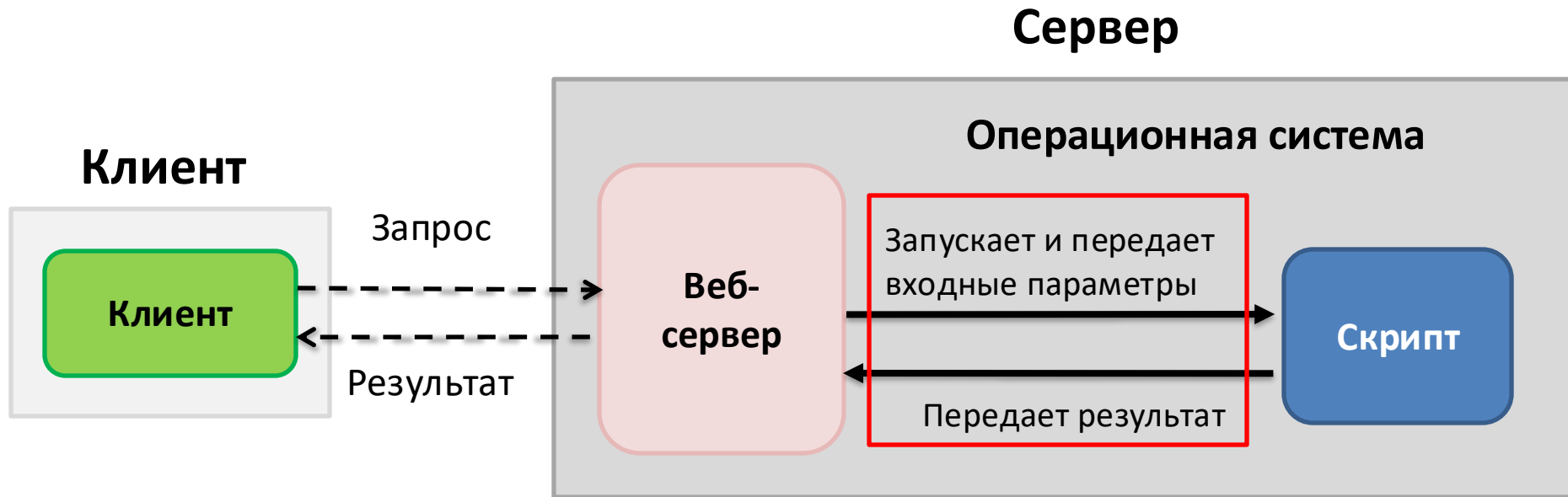
# Веб-приложения



# Веб-приложение/динамическая страница

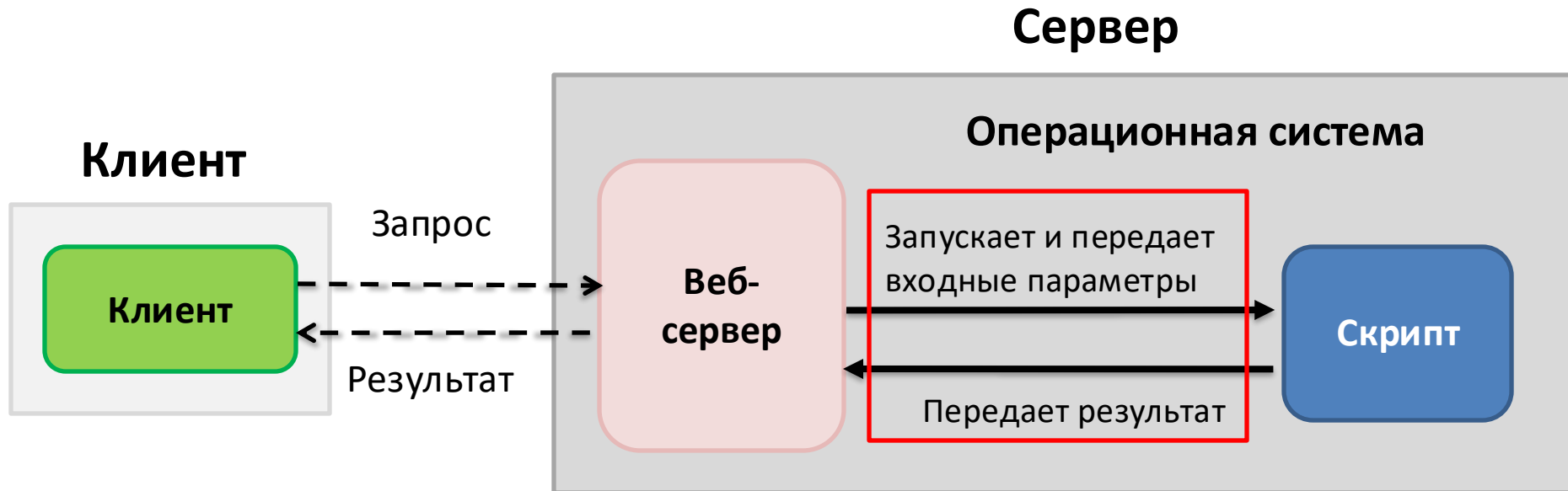


# Веб-приложение/динамическая страница



- Как веб-сервер запускает скрипт и передает ему входные параметры?
- Как скрипт передает результаты своей работы веб-серверу?

# Веб-приложение/динамическая страница



- Как веб-сервер запускает скрипт и передает ему входные параметры?
- Как скрипт передает результаты своей работы веб-серверу?

Исторически первый и самый простой - стандарт CGI, соответствующий механизму работы консольных приложений в серверной операционной системе.

# Вопросы

- Что такое консольное приложение?
- Какие операционные системы могут быть на сервере?
- Как запускаются и работают консольные приложения?
- Зависят ли эти механизмы от языка, на котором написано приложение, и от операционной системы?
- Как в консольные приложения передаются входные параметры?
- Куда и как выводятся результаты работы консольных приложений?

# Некоторые ответы

- Что такое консольное приложение? Приложение, выполняющееся в консоли:)
- Какие операционные системы могут быть на сервере? Linux (почти наверняка), Windows или FreeBSD
- Как запускаются и работают консольные приложения? Консольные приложения запускаются в терминале и выполняются командной оболочкой
- Зависят ли эти механизмы от языка, на котором написано приложение, и от операционной системы? Общие механизмы работы с консольными приложениями в Linux и Windows похожи, но есть нюансы

# **Серверная операционная система**

| Category                | Source                                    | Date      | Linux  | UNIX and Unix-like (not incl. Linux)  | Windows  | In-house                  | Other  |
|-------------------------|---|-----------|--|---|--|---------------------------|--------|
| Desktop, laptop         | StatCounter Global Stats <sup>[271]</sup> | June 2023 | Linux kernel family 7.23%:<br>ChromeOS 4.15% (in the US up to 8.0%) plus traditional "Linux" 3.08%         | 21.38% (macOS)  | 68.15% (all versions)  |                           | 3.24%  |
| Embedded <sup>[e]</sup> | EE Times <sup>[272]</sup>                 | Mar 2019  | 38.42% (embedded Linux, Ubuntu, Android, other)  | 2.82% (QNX, LynxOS)   | 10.73% (Windows 10, Windows Embedded Compact)  | 10.73%                    | 37.30% |
| Mainframe               | Gartner <sup>[262]</sup>                  | Dec 2008  | 28% (SLES, RHEL)   |   |  | 72% (z/OS) <sup>[f]</sup> |        |
| Server (web)            | W3Techs <sup>[273]</sup>                  | Sep 2021  | Likely 77.4% (39.8% confirmed) <sup>[g]</sup> (Ubuntu, CentOS, Debian, Gentoo, RHEL, ...) <sup>[274]</sup> | Less than 1% is confirmed to be UNIX or Unix-like and non-Linux. The top operating systems in order are: 0.3% BSD (97.8% of which is FreeBSD), <sup>[275]</sup> <0.1% Darwin, <sup>[276]</sup> <0.1% HP-UX, <sup>[277]</sup> <0.1% Solaris, <sup>[278]</sup> and <0.1% Minix. <sup>[279][g]</sup> | 22.7% (Windows Server 2019, WS2016, WS2012)<br>Microsoft's own webserver runs 6.6% of websites. <sup>[280]</sup> |                           |        |
| Smartphone, tablet      | StatCounter Global Stats <sup>[281]</sup> | Apr 2020  | 70.80% (Android, KaiOS)  | 28.79% (iOS)  | 0.07%  |                           | 0.34%  |
| Supercomputer           | TOP500 <sup>[282]</sup>                   | Nov 2019  | 100% (Custom)  |   |  |                           |        |

Кто и когда создал операционную систему Linux?

# Linux



Линус Торвальдс (родился в 1969 Финляндии)  
**1991** год

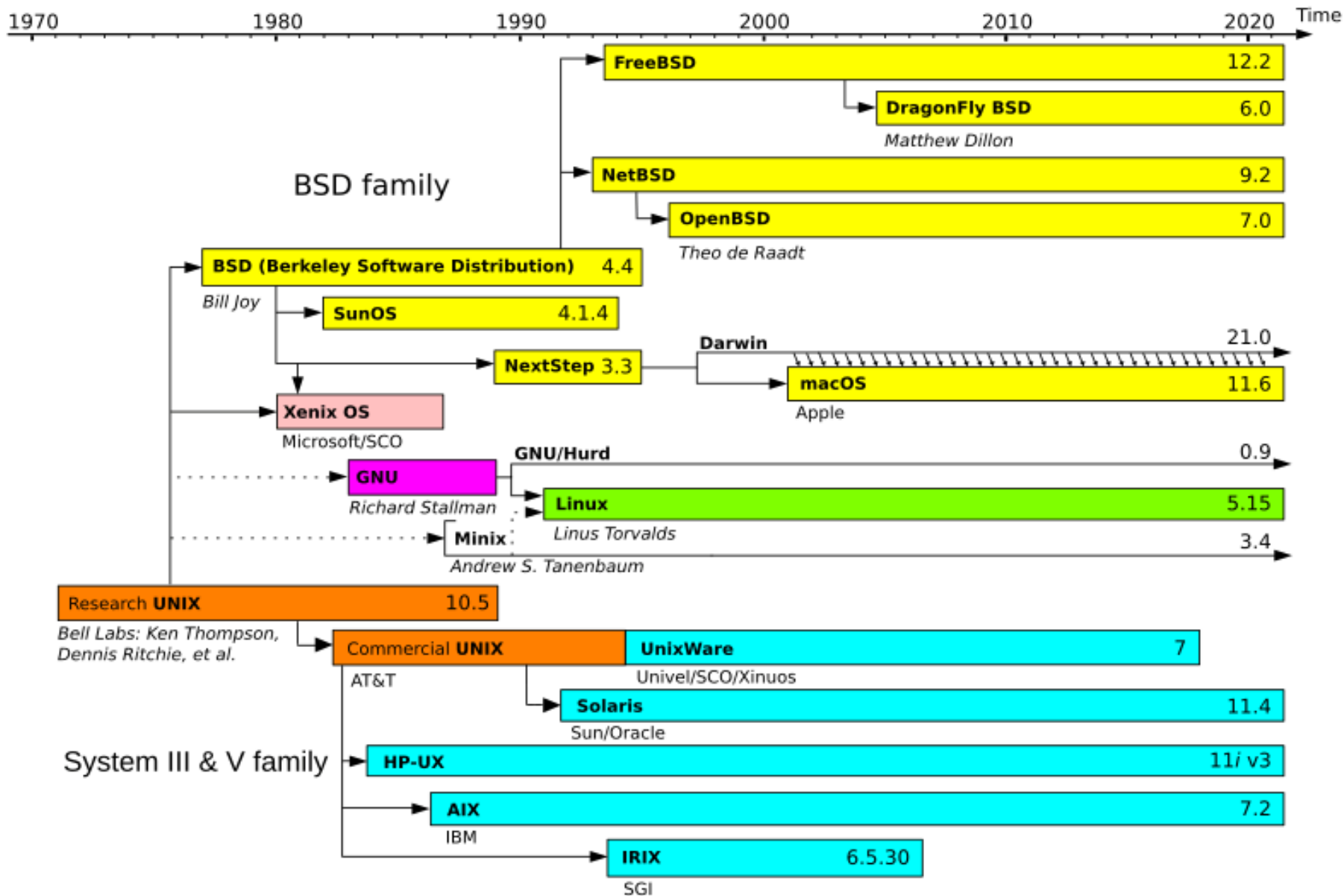
От: torvalds@klaava.Helsinki.Fi (Линус Бенедикт Торвальдс) Новостная группа: comp.os.minix  
Тема: Небольшой опрос о моей новой операционной системе  
Дата: 25 Aug 91 20:57:08 GMT Организация: Хельсинкский Университет

Привет всем тем, кто использует minix —

Я делаю (свободную) операционную систему (это только хобби, не столь большое и профессиональное, как GNU) для 386(486)AT клонов. ... Я хочу получить любой отзыв, касающийся вещей, которые нравятся/не нравятся людям в minix, так как моя ОС похожа на неё...

PS. Да, в ней нет кода minix, и будет мультипоточковая ФС. Система НЕПЕРЕНОСИМА (использует команды Intel 386 и т.д.) и, вероятно, будет поддерживать только жесткие диски AT, так как это всё, что у меня есть :-)

# История развития и распространения Unix



# Эволюция компьютеров и ОС

Mainframe Computing  
1960s



Mini Computing  
1970s



Personal Computing  
1980s



Desktop Internet Computing  
1990s



Mobile Internet Computing  
2000s



**UNIX**

**Linux**

**MS DOS Windows**

## Терминал - оконечное устройство



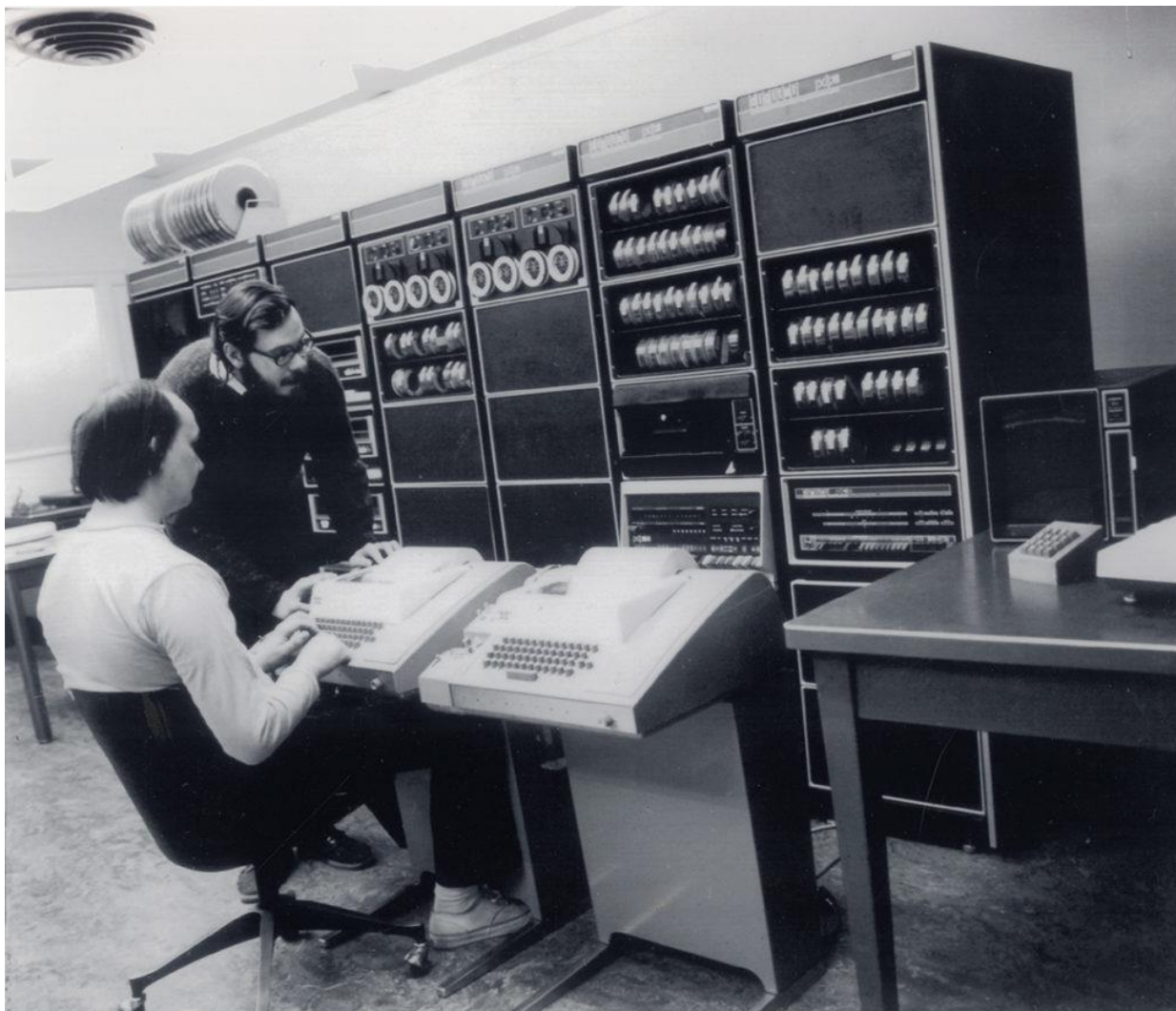
Первые терминалы - телетайпы  
(teletypewriters, TTY)

Текст



Мейнфреймы и миникомпьютеры

<https://youtu.be/9TGQ4pnVWSQ?t=9>



Создатели операционной системы UNIX за терминалом:  
Кен Томпсон (слева, в очках) и Деннис Ритчи

**Консоль** - устройство (не интеллектуальное) со встроенной клавиатурой и монитором



Терминал - программа внутри консоли

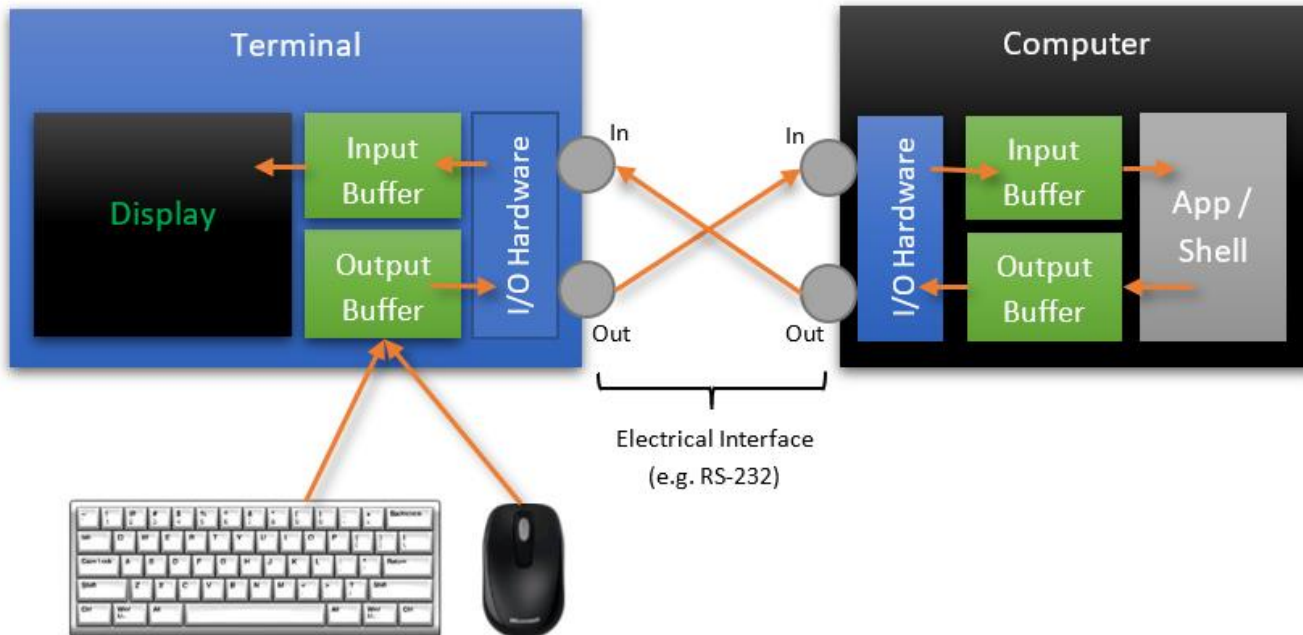


### Консольное приложение:

- запускается с помощью терминала из консоли
- выводит результаты в консоль

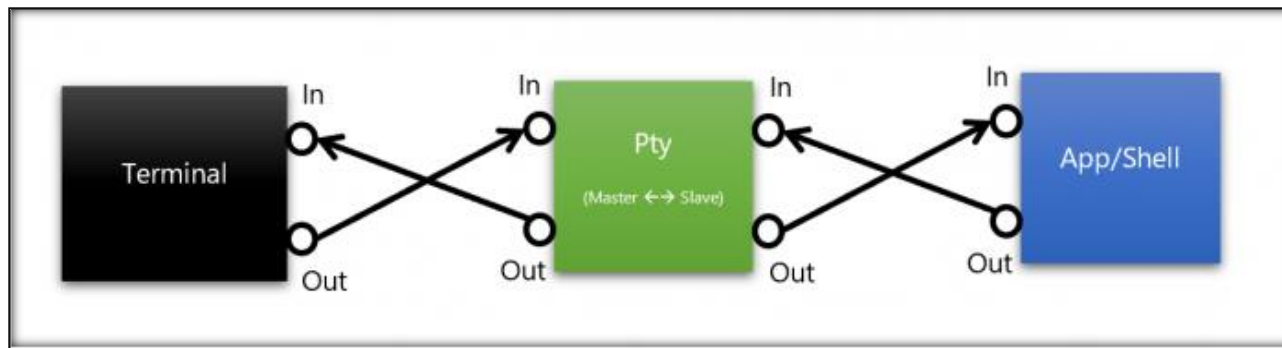
# Задачи терминала

- Распознавание и прием символов, набираемых на клавиатуре.
- Формирование из принимаемых символов единой строки с учетом управляющих кодов.
- Обмен текстом с компьютером с помощью коммуникационного оборудования через прямое физическое соединение или по линиям связи.
- Отображение полученного от компьютера текста на дисплее.



# Терминал и консоль

В случае ПК это синонимы, обозначающие программу-эмулятор физического устройства с возможностями редактирования



```
C:\WINDOWS\system32\cmd.exe
Microsoft Windows [Version 10.0.19041.572]
(c) Корпорация Майкрософт (Microsoft Corporation), 2020. Все права защищены.

C:\Users\andrv>
```

## Командные оболочки (shells)

- Принимают и парсят строки от терминала, выполняют действия сами или запускают другие приложения, возвращают строки символов терминалу.
- Не имеют своего UI, работают через терминал.
- Могут быть с терминалом на одной или на разных машинах.

# Терминалы и оболочки

| Операционная система | Терминалы   | Оболочки                      |
|----------------------|---|-------------------------------|
| Windows              | ConHost (командная строка)<br>Windows Terminal<br>ConEme, Cmder | cmd<br>PowerShell<br>Git Bash |
| Linux/MacOS<br>WSL   | Terminal<br>iTerm2<br>Alacritty                                 | bash<br>zsh<br>fish           |

# Рождение UNIX



Кен Томпсон (р. 1943)

Деннис Ритчи (1941-2011)

Исследовательская лаборатория Bell Labs компании AT&T,  
**1970** год

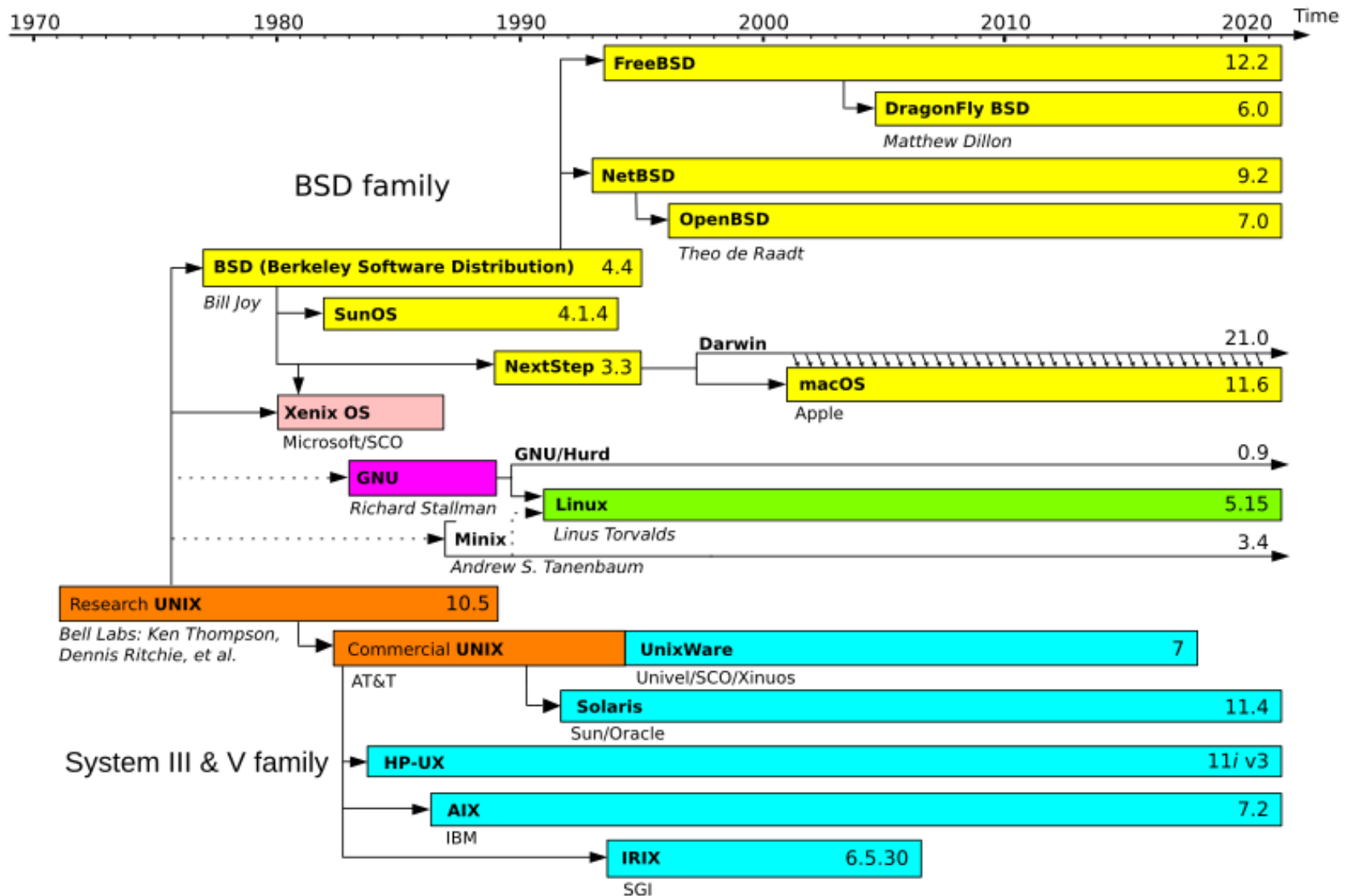
# Операционная система UNIX

---

Мультиплатформенная мультизадачная многопользовательская ОС

- "Всё есть файл" и "всё есть текст"
  - Настройка и управление системой через текстовые файлы.
  - Представление физических и виртуальных устройств в виде файлов.
- Интерактивная работа
  - Взаимодействие с пользователем посредством терминала.
  - Командные оболочки и набор утилит командной строки для управления файловой системой и обработки текста.
  - Стандартные потоки ввода/вывода и конвейеры, позволяющие вывод одной консольной программы направлять на вход другой.

# Распространение Unix в 1970-1980 гг



# Стандартизация Unix

POSIX (Portable Operating System Interface) - стандарт (с 1988 года), описывающий интерфейс между ОС и прикладной программой, библиотеку языка C и набор приложений и их интерфейсов.

## **Цель создания:**

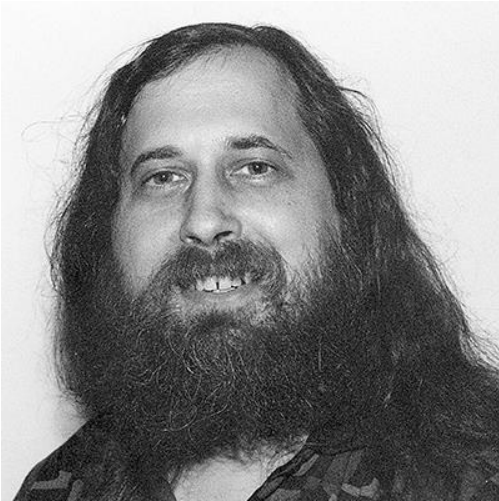
- совместимость Unix-like операционных систем;
- переносимость программ на уровне исходного кода.

## **Признаки:**

- разграничение прав пользователей и групп;
- суперпользователь root с привилегированными правами;
- древовидная файловая система с единым корнем /;
- настройка система и программ через текстовые файлы;
- единый API программирования языка C;
- единый стандарт консольных утилит и команд (POSIX 2).

# GNU (GNU's Not Unix)

---



Ричард Столлман (р. 1953)  
Разработчик Emacs, gcc, gdb

**1983** год

Цель - создание полностью открытой Unix-совместимой ОС со всеми необходимыми утилитами

Лицензия на свободное ПО (GNU General Public License):

- Исходный код свободен для изучения
- Свободно распространяется
- Свободно изменяется
- Модификация свободно распространяется

## **GNU (GNU's Not Unix)**

---

К началу 1990-х была написана оболочка `bash` и большинство стандартных утилит командной строки.

Ядро операционной системы `Hurd` не было готово.

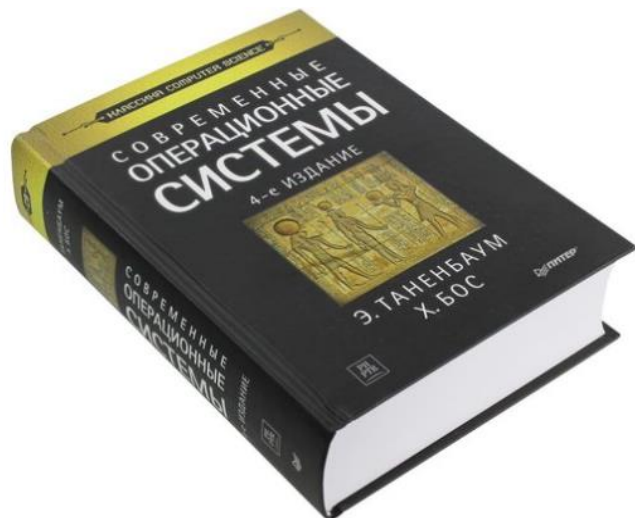
# Minix



Эндрю Танненбаум (р. 1944)  
Амстердамский свободный университет,  
**1987** год

Операционная система с открытым кодом  
– учебное пособие к книге.

В Usenet была создана группа новостей  
comp.os.minix для обсуждения Minix, много  
людей были заинтересованы в развитии  
свободной UNIX-совместимой  
операционной системы.



# GNU/Linux

---



Линус Торвальдс (родился в 1969 в Финляндии)  
**1991** год

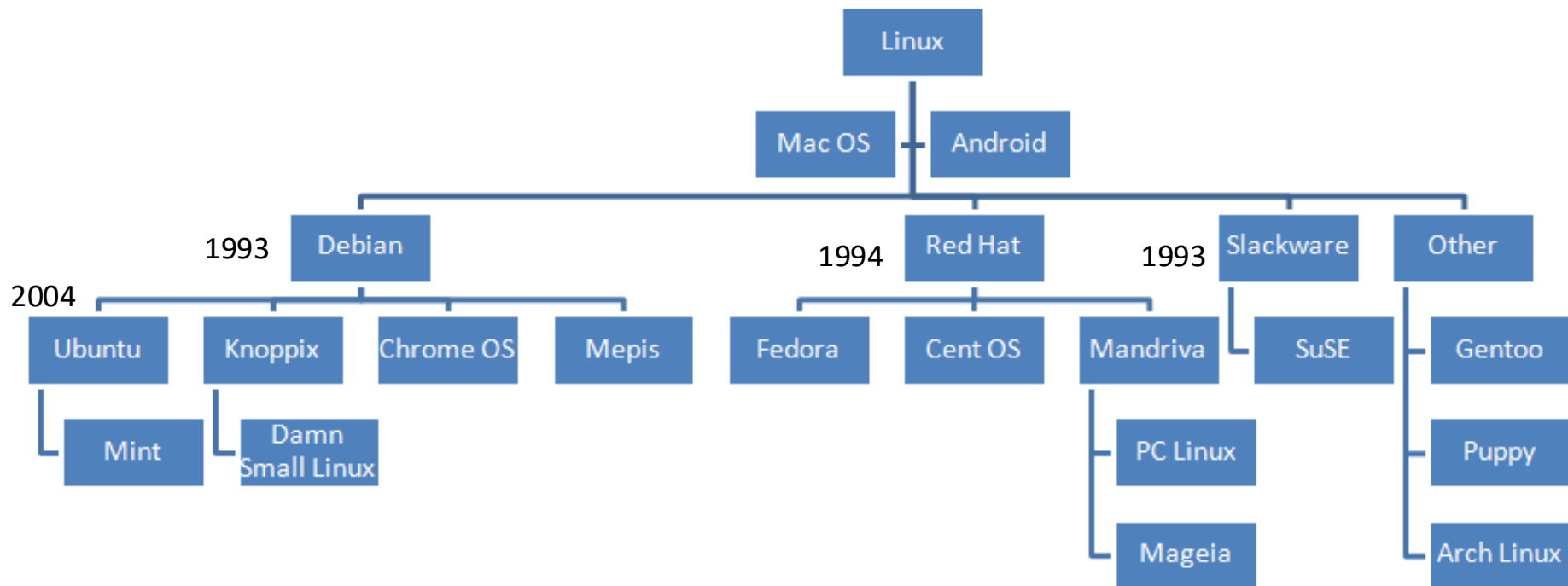
От: torvalds@klaava.Helsinki.Fi (Линус Бенедикт Торвальдс) Новостная группа: comp.os.minix  
Тема: Небольшой опрос о моей новой операционной системе  
Дата: 25 Aug 91 20:57:08 GMT Организация: Хельсинкский Университет

Привет всем тем, кто использует **minix** —

Я делаю (свободную) операционную систему (это только хобби, не столь большое и профессиональное, как **GNU**) для 386(486)AT клонов. ... Я хочу получить любой отзыв, касающийся вещей, которые нравятся/не нравятся людям в minix, так как моя ОС похожа на неё...

PS. Да, в ней нет кода minix, и будет мультипоточковая ФС. Система НЕПЕРЕНОСИМА (использует команды Intel 386 и т.д.) и, вероятно, будет поддерживать только жесткие диски AT, так как это всё, что у меня есть :-)

# Дистрибутивы, рост популярности



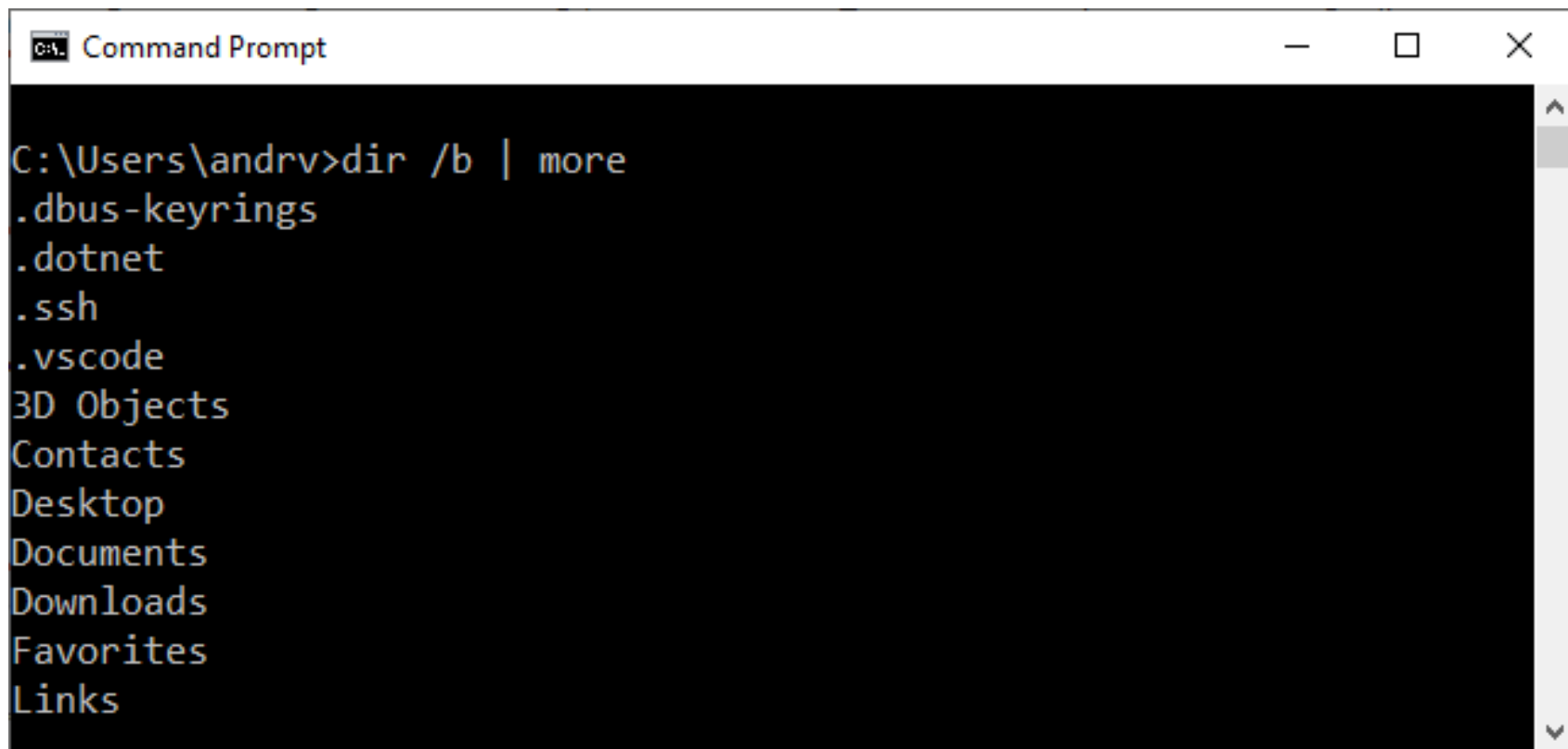
- С 1993 года используют профессиональные разработчики, студенты
- С 1995 года Linux выходит в корпоративный сегмент. Веб-провайдеры стали использовать Linux для хостинга. Стандартом стал LAMP (Linux + Apache + MySQL + PHP)

# Консольные приложения



# Режимы работы с терминалом

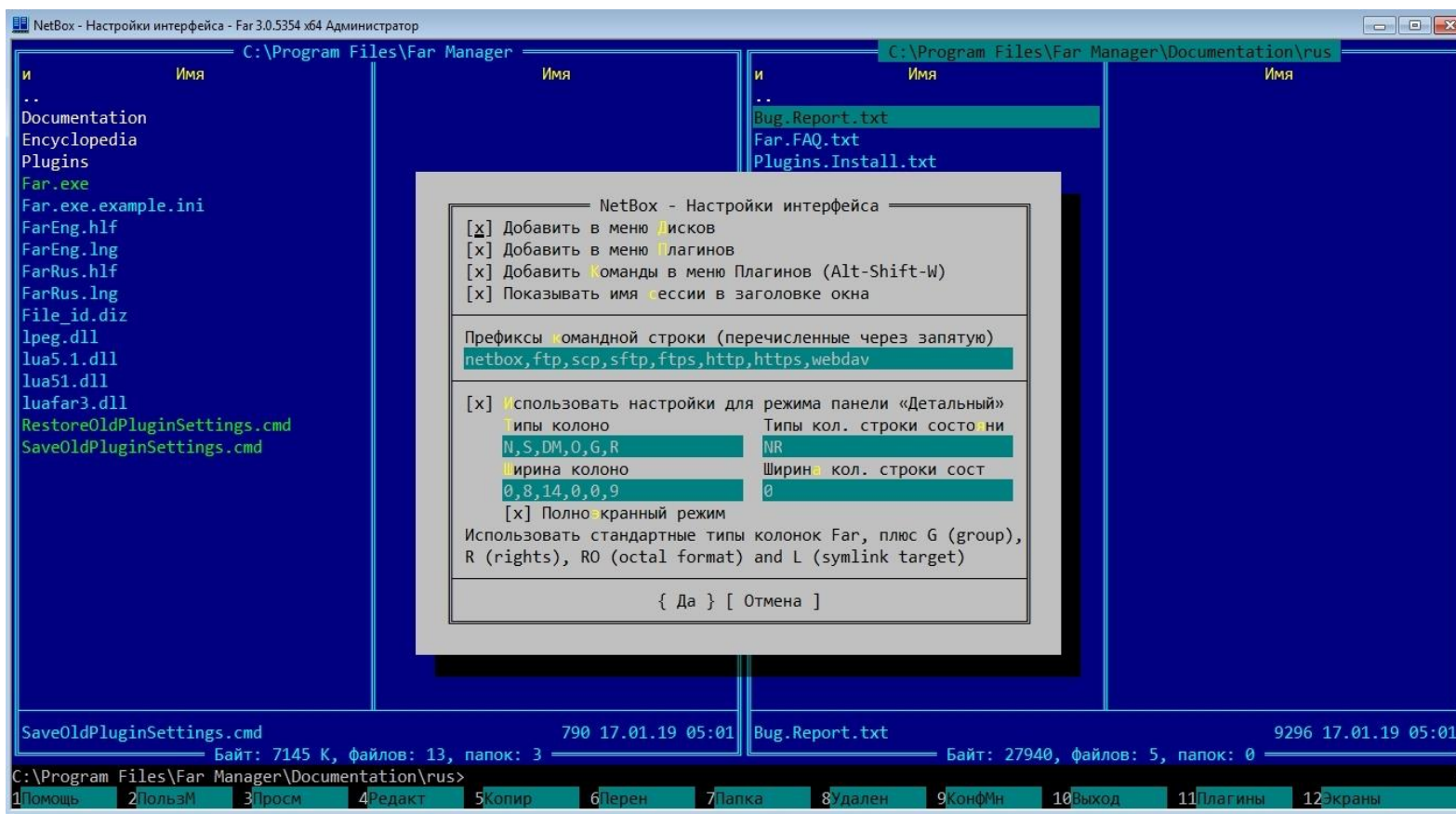
**Канонический потоковый ввод/вывод** - терминал поддерживает буфер ввода с возможностями редактирования



```
C:\Users\andrv>dir /b | more
.dbus-keyrings
.dotnet
.ssh
.vscode
3D Objects
Contacts
Desktop
Documents
Downloads
Favorites
Links
```

# Режимы работы с терминалом

**Сырой (raw) режим** - терминал передает приложению все символы, введенные пользователем, напрямую. Приложение их обрабатывает самостоятельно.



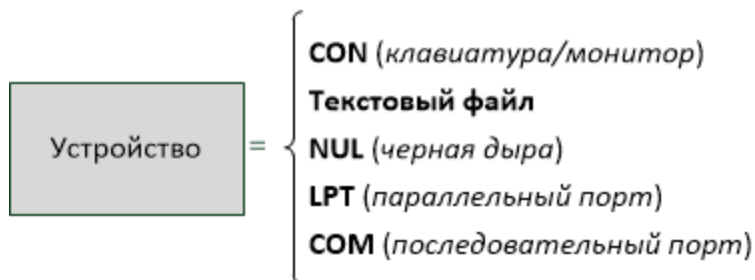
# Консольные полноэкранные приложения

```
86
87 func (c *Color) prepend(value Attribute) {
88     c.params = append(c.params, 0)
89     copy(c.params[1:], c.params[0:])
90     c.params[0] = value
91 }
92
93 // Output defines the standard output of the print functions. By default
94 // os.Stdout is used.
95 var Output io.Writer = os.Std
96     var Stdin *os.File
97 // Print formats using the its operands and writes to
98 // standard output. Space Stderr *os.File its operands and writes to
99 // string. It returns the number of bytes written and any write error
100 // encountered. This is the standard fmt.Print() method wrapped with the given
101 // color.
102 func (c *Color) Print(a ...interface{}) (n int, err error) {
103     c.Set()
104     defer Unset()
105
106     return fmt.Fprint(Output, a...)
107 }
108
109 // Printf formats according to a format specifier and writes to standard output.
110 // It returns the number of bytes written and any write error encountered.
111 // This is the standard fmt.Printf() method wrapped with the given color.
112 func (c *Color) Printf(format string, a ...interface{}) (n int, err error) {
113     c.Set()
114     defer Unset()
115
116     return fmt.Fprintf(Output, format, a...)
117 }
118
119 // Println formats using the default formats for its operands and writes to
120 // standard output. Spaces are always added between operands and a newline is
121 // appended. It returns the number of bytes written and any write error
122 // encountered. This is the standard fmt.Print() method wrapped with the given
123 // color.
124 func (c *Color) Println(a ...interface{}) (n int, err error) {
INSERT master color.go[+] Output < go utf-8[unix] 33% : 95: 30 Tagbar Name color.go
79
80 var {
81     Stdin = NewFile(uintptr(syscall.Stdin), "/dev/stdin")
82     Stdout = NewFile(uintptr(syscall.Stdout), "/dev/stdout")
83     Stderr = NewFile(uintptr(syscall.Stderr), "/dev/stderr")
84 }
85 Stdin, Stdout, and Stderr are open Files pointing to the standard input,
86 standard output, and standard error file descriptors.
87
```

```
▼ variables
+Output : io.Writer
+Attribute : int
▼+Color : struct
[fields]
-params : []Attribute
[methods]
+Add(value ) : *Color
+Print(a ) : int, error
+PrintFunc() : func(a )
+Printf(format string, a ) : int, err
+PrintfFunc() : func(format string, a
+Println(a ) : int, error
+PrintlnFunc() : func(a )
+Set() : *Color
+SprintfFunc() : func(a ) string
+SprintfFunc() : func(format string,
+SprintfFunc() : func(a ) string
-format() : string
-prepend(value Attribute)
-sequence() : string
-unformat() : string
-wrap(s string) : string
[functions]
+New(value ) : *Color
+Set(p ) : *Color
▼ functions
+Black(format string, a )
+BlackString(format string, a ) : str
+Blue(format string, a )
+BlueString(format string, a ) : stri
+Cyan(format string, a )
+CyanString(format string, a ) : stri
+Green(format string, a )
+GreenString(format string, a ) : str
```

```
[Godoc] [-] godoc [unix] 14% : 84: 1
-- INSERT --
```

# Потоки ввода/вывода (Windows)



echo Привет! > сору.txt

хсору /? >> хсору\_help.txt

хсору D:\1.txt C: > сору.txt **2>&1**

date < date.txt

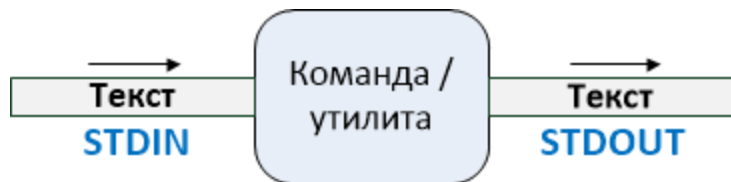
# Вывод в файл (перезапись)

# Добавление в файл

# Дублирование ошибок в STDOUT

# Ввод из файла

# Конвейеризация команд



Одна команда = одна функция  
find, cat, sed, grep, awk, ...



Командный язык для  
поддержки программных  
конвейеров

**cat** меню | **grep** креветки | **test -lt \$10**

```
find /tmp -type f -name '*' -mtime +7 -print0 | xargs -0 rm -f
```



# Синтаксис командной строки

---

```
find . -type f -name '*.txt' -exec grep -Hni 'search_term' {} \;
```

- `find .` — запустить поиск в текущем каталоге и всех его подкаталогах
- `-type f` — оставить только файлы
- `-name '*.txt'` — выбирать только файлы с расширением txt
- `-exec grep -Hni 'search_term' {} \;` — вызывает grep для каждого найденного файла. Символ `{}` заменяется именем файла, а `;` завершает команду.
  - `-H` — вывести имена файлов.
  - `-n` — вывести номера строк.
  - `-i` — игнорировать регистр при поиске.
  - `'search_term'` — строка, которую мы ищем.

# Переменные окружения

```
→ ~ env | sort
```

```
HOME=/home/andrey
```

```
HOSTTYPE=x86_64
```

```
LANG=C.UTF-8
```

```
LESS=-R
```

```
LOGNAME=andrey
```

```
LSCOLORS=Gxfxcxdxbxegedabagacad
```

```
LS_COLORS=rs=0:di=01;34:ln=01;36:mh=00:pi=40;33:so=01;35:do=01;35:bd=40;33;01:cd=40;33;01:or=40;31;01:mi=00:su=37;41:sg=30;43:ca=30;41:tw=30;42:ow=34;42:st=37;44:ex=01;32:*.tar=01;31:*.tgz=01;31:*.arc=01;31:*.arj=01;31:*.taz=01;31:*.lha=01;31:*.lz4=01;31:*.lzh=01;31:*.lzma=01;31:*.tlz=01;31:*.txz=01;31:*.tzo=01;31:*.t7z=01;31:*.zip=01;31:*.z=01;31:*.Z=01;31:*.dz=01;31:*.gz=01;31:*.lrz=01;31:*.lz=01;31:*.lzo=01;31:*.xz=01;31:*.zst=01;31:*.tzst=01;31:*.bz2=01;31:*.bz=01;31:*.tbz=01;31:*.tbz2=01;31:*.tz=01;31:*.deb=01;31:*.rpm=01;31:*.jar=01;31:*.war=01;31:*.ear=01;31:*.sar=01;31:*.rar=01;31:*.alz=01;31:*.ace=01;31:*.zoo=01;31:*.cpio=01;31:*.7z=01;31:*.rz=01;31:*.cab=01;31:*.wim=01;31:*.swm=01;31:*.dwm=01;31:*.esd=01;31:*.jpg=01;35:*.jpeg=01;35:*.mjpg=01;35:*.mjpeg=01;35:*.gif=01;35:*.bmp=01;35:*.pbm=01;35:*.pgm=01;35:*.ppm=01;35:*.tga=01;35:*.xbm=01;35:*.xpm=01;35:*.tif=01;35:*.tiff=01;35:*.png=01;35:*.svg=01;35:*.svgz=01;35:*.mng=01;35:*.pcx=01;35:*.mov=01;35:*.mpg=01;35:*.mpeg=01;35:*.m2v=01;35:*.mkv=01;35:*.webm=01;35:*.ogm=01;35:*.mp4=01;35:*.m4v=01;35:*.mp4v=01;35:*.vob=01;35:*.qt=01;35:*.nuv=01;35:*.wmv=01;35:*.asf=01;35:*.rm=01;35:*.rmvb=01;35:*.flc=01;35:*.avi=01;35:*.fli=01;35:*.flv=01;35:*.gl=01;35:*.dl=01;35:*.xcf=01;35:*.xwd=01;35:*.yuv=01;35:*.cgm=01;35:*.emf=01;35:*.ogv=01;35:*.ogx=01;35:*.aac=00;36:*.au=00;36:*.flac=00;36:*.m4a=00;36:*.mid=00;36:*.midi=00;36:*.mka=00;36:*.mp3=00;36:*.mpc=00;36:*.ogg=00;36:*.ra=00;36:*.wav=00;36:*.oga=00;36:*.opus=00;36:*.spx=00;36:*.xspf=00;36:
```

```
NAME=DESKTOP-BU86I5T
```

```
NVM_DIR=/home/andrey/.nvm
```

```
OLDPWD=/home/andrey
```

```
PAGER=less
```

```
PATH=/home/andrey/.composer/vendor/bin:/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin:/usr/games:/usr/local/games:/usr/lib/wsl/lib:/mnt/c/Python39/Scripts:/mnt/c/Python39:/mnt/c/WINDOWS/system32:/mnt/c/WINDOWS:/mnt/c/WINDOWS/System32/Wbem:/mnt/c/WINDOWS/System32/WindowsPowerShell/v1.0:/mnt/c/WINDOWS/System32/OpenSSH:/mnt/c/ProgramData/ComposerSetup/bin:/mnt/c/Program Files/Microsoft VS Code/bin:/mnt/c/Program Files/Git/cmd:/mnt/c/Program Files/nodejs:/mnt/c/ProgramData/chocolatey/bin:/mnt/c/SQLite:/mnt/c/Users/andr/OneDrive/Far3x64:/mnt/c/PHP8:/mnt/c/Users/andr/AppData/Local/Microsoft/WindowsApps:/mnt/c/Users/andr/AppData/Roaming/Composer/vendor/bin:/mnt/c/Users/andr/AppData/Roaming/npm:
```

```
PWD=/home/andrey
```

# Переменные окружения в Linux

- Создать свою переменную и присвоить ей значение можно с помощью команды `export`.

```
export MYVAR=hello
```

- Можно передавать переменные в приложение через командную строку или команду `env`.

```
NAME=value команда
```

```
env NAME=value ... команда аргументы
```

# Переменные окружения в Windows

- Создать свою переменную и присвоить ей значение можно с помощью команды `set`.

```
set MYVAR=hello
```

Переменная создается в текущем сеансе, доступна для всех команд и приложений.

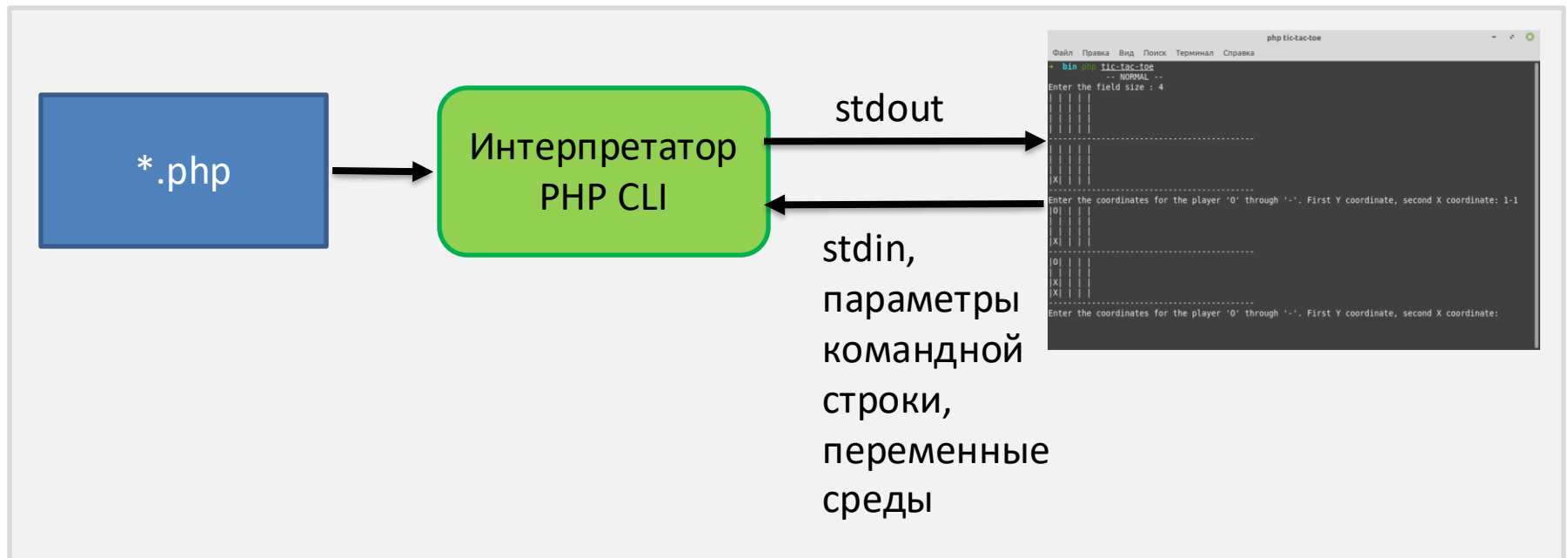
- Одноразовая передача переменных в приложение:

```
set NAME=value && команда
```

```
$env:MY_VARIABLE = "my_value"; ./my_program.exe
```

# Запуск консольного приложения на PHP

> MYVAR=hello php my\_script.php arg1 arg2



```
$s = readline("Введите строку: "); // Чтение строки из stdin
echo "Вы ввели: $s\n";           // Вывод строки в stdout
```

## Потоки php://std\*

- Прямой доступ к потокам PHP-процесса:  
`php://stdin`, `php://stdout`, `php://stderr`
- Константы `STDIN`, `STDOUT`, `STDERR` позволяют обращаться к уже открытым потокам (доступны только в CLI-режиме).

# Запуск консольного приложения на PHP

```
$ MYVAR=hello ./my_script.php arg1 arg2
```

Чтобы сделать скрипт исполняемым файлом, нужно:

- Установить для него флаг x (execute)  
`chmod +x ./my_script.php`
- Записать в начало скрипта путь к интерпретатору (шебанг)

```
#!/usr/bin/php
```

или

```
#!/usr/bin/env php
```

Если каталог для скрипта добавить в переменную PATH, можно будет запускать его без указания пути.

```
$ MYVAR=hello my_script.php arg1 arg2
```

## Вопросы и ответы

- Как запускаются консольные приложения? Команда для запуска вводится в терминале, приложения запускает командная оболочка.
- Зависят ли эти механизмы от языка, на котором написано приложение? Не зависят, любой скрипт, для которого установлен интерпретатор, можно сделать исполняемым файлом.
- Как в консольные приложения передаются входные параметры? Через поток `stdin`, аргументы командной строки и переменные среды.
- Куда и как выводятся результаты работы консольных приложений? В поток `stdout`. Сообщения об ошибках можно выводить в `stderr`.