

С и Unix

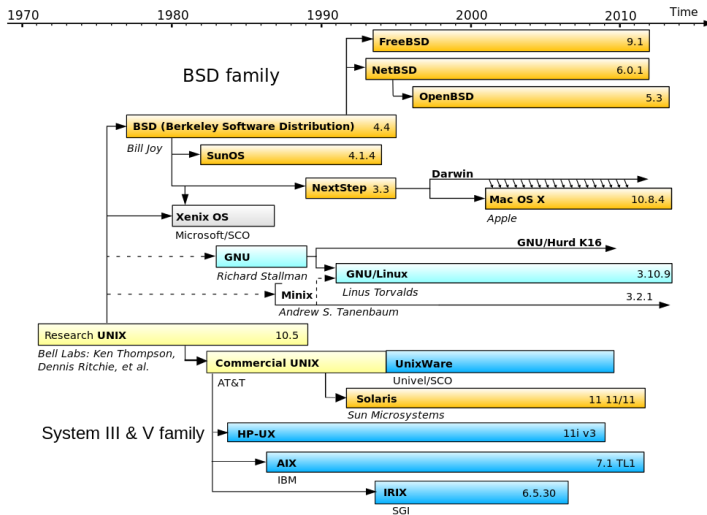
Unix – операционная система разработанная в 1969 году в лаборатории AT&T группой людей среди которых: Ken Thompson, Dennis Ritchie и Brian Kernighan.



Сейчас под Unix понимают разные операционные системы удовлетворяющие «стандартам Unix».

👉 В ходе разработки Unix был создан язык С

История Unix



Операционная система Unix

- многопользовательская, многозадачная операционная система
- для программирования, обработки текстов, комуникации
- используется как сервера: mail, web, print ...

Средства поддержки

- ✓ Компиляторы: C, C++, Python, Java, Fortran ...
- ✓ Командные оболочки: bash, zsh, csh, tcsh ...
- ✓ Обработка текста: cat, grep, sed, awk, diff ...
- ✓ Коммуникации: ssh, mail, ftp ...

Философия UNIX:

- программа делает что-то одно и делает это хорошо
- программы должны работать вместе и использовать текстовые потоки, поскольку это универсальный интерфейс

Работа в OS Unix

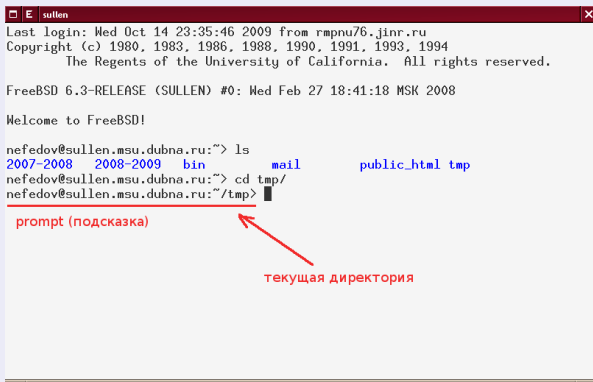
Доступ к Unix машине

- Вам необходима «учётная запись» (a user account)
- Сидя перед терминалом набираете «user name», «password» и заходите на машину
- Имеется возможность удалённой работы на машине: `ssh`, `telnet` ...

Командная строка:

Взаимодействие с системой осуществляется набором команд в командной оболочке, в так называемом «**shell**»

Shell (sh) — интерпретатор, позволяющий выполнять **скрипты** — списки команд сохраненные в файле



```
sullen
Last login: Wed Oct 14 23:35:46 2009 from rmpnu76.jinr.ru
Copyright (c) 1980, 1983, 1986, 1988, 1990, 1991, 1993, 1994
    The Regents of the University of California.  All rights reserved.

FreeBSD 6.3-RELEASE (SULLEN) #0: Wed Feb 27 18:41:18 MSK 2008

Welcome to FreeBSD!

nefedov@sullen.msu.dubna.ru:~> ls
2007-2008  2008-2009  bin          mail          public_html tmp
nefedov@sullen.msu.dubna.ru:~> cd tmp/
nefedov@sullen.msu.dubna.ru:~/tmp>
```

prompt (подсказка)

текущая директория

Файлы и папки (директории)

- Файлы и папки хранятся в файловой системе
- Папки организованы в виде иерархической древовидной системы
- Начальная (корневая: **root**) точка файловой системы – `/`
- Знак `/` разделяет один уровень от другого; сравните с `\` в windows
- Пример: `/usr/bin` – папка **bin** находящаяся в папке **usr** которая лежит в «корне» `/`
- Домашняя (пользовательская) папка обычно находится в `/home/username`, а также доступна через обозначение `~username/`

Некоторые полезные команды

<code>ls <dir></code>	содержимое (список,лист) папки
<code>cd <dir></code>	перейти в папку <code><dir></code>
<code>mkdir <dir></code>	создать новую папку <code><dir></code>
<code>pwd</code>	имя текущей папки

<code>cp <file> <newfile></code>	скопировать файл
<code>mv <file> <newfile></code>	переименовать (переместить) файл
<code>rm <file></code>	удалить файл
<code>less <file></code>	просмотр файла (поиск и т.п.)
<code>cat <f1> <f2> ...</code>	«распечатать» файлы

Структура команды

`command` `[-options]` `arguments`

- ✓ **Имя команды**
- ✓ **Модификатор команды:** обычно первый символ минус и затем имя опции (без пробелов)
- ✓ **Аргументы:** имена файлов, имена объектов ...

Внимание

- UNIX чувствителен к использованию строчных или заглавных букв в именах и командах
- Пробелы разделяют команду, опции и аргументы

Примеры:

- Перейти в домашнюю папку:
`cd` или `cd ~` или `cd $HOME`
- Создать папку «Petrov»: `mkdir Petrov`
Посмотреть список файлов: `ls`
`bin public_html Petrov mail tmp`
- Перейти в папку «Petrov»: `cd Petrov`
Посмотреть имя текущей папки: `pwd`
`/home/nefedov/Petrov`
- Перейти на один уровень ниже: `cd ../; pwd`
`/home/nefedov`

Read The Fine Manual: man command

- Команды в Unix имеют описания доступные с помощью программы **man** (manual, руководство)
- Пример: **> man ls**

```
LS(1)                                User Commands                                LS(1)

NAME
  ls - list directory contents

SYNOPSIS
  ls [OPTION]... [FILE]...

DESCRIPTION
  List information about the FILES (the current directory by default).
  Sort entries alphabetically if none of -cftuvSUX nor --sort.

  Mandatory arguments to long options are mandatory for short options
  too.

  -a, --all
        do not ignore entries starting with .

  -A, --almost-all
        do not list implied . and ..

Manual page ls(1) line 1
```

- Если имя программы или команды неизвестно, нужно использовать:
> man -k 'образец для поиска' (например: **man -k print**)

Редакторы текста

- GNU (x)emacs – gui/text редактор или даже «среда редактирования». Широко применяется и имеет множество поклонников. Документация:
 - `menu emacs -> Help -> Emacs Tutorial`
 - `http://www.gnu.org/manual/manual.html`
- (g)vim – наиболее мощный редактор, но работает на совершенно иных принципах. Труден для начального освоения. Документация:
 - `vimtutor`
 - Книга с описанием (English):
`ftp://ftp.vim.org/pub/vim/doc/book/vimbook-OPL.pdf`
 - `http://www.vim.org`
- jEdit – кросс-платформенный (Java) редактор
- gedit, kate, nedit, pico, joe, mcedit ... – «маленькие» редакторы

Первая программа

- Запускаем редактор: `emacs &` (обратите внимание на `&`)

```
/* Hello World in C */  
#include <stdio.h>  
  
int main() {  
    printf("Hello, world!\n");  
}
```

- Компиляция: `clang hello.c` (или `gcc hello.c`)
- Что получилось? `ls -l`

<code>-rwxr-xr-x</code>	<code>1</code>	<code>nefedov</code>	<code>nefedov</code>	<code>4555</code>	<code>Сен</code>	<code>1</code>	<code>00:00</code>	<code>a.out</code>
<code>-rw-r--r--</code>	<code>1</code>	<code>nefedov</code>	<code>nefedov</code>	<code>76</code>	<code>Сен</code>	<code>1</code>	<code>00:00</code>	<code>hello.c</code>
- Выполнение: `./a.out` (обратите внимание на `./`)
`Hello, world!`

Перенаправление потоков

- `stdout` → файл

```
./program > output.dat # записать вывод прог. в файл output.dat
```

```
./program >> output.dat # добавить вывод программы в конец файла
```

- файл → `stdin`

```
./program < input.dat # чтение из файла input.dat
```

- чтение из файла `input.dat` и вывод в файл `output.dat`

```
./program < input.dat > output.dat
```

- 🔗 дескрипторы файлов: 0, 1 и 2 соответствуют `stdin`, `stdout` и `stderr`

```
./program 0< finput 1> foutput # тоже, что предыдущая строка
```

- `stdout` → файл1, `stderr` → файл2

```
./program > out.dat 2> outerr.dat
```

- `stdout` + `stderr` → файл

```
./program > output.dat 2>&1 # 2(=stderr) выводится в 1(=stdout)
```

```
./program >& output.dat # simplified non-POSIX
```

Конвейер (pipe)

program1 | program2

☞ вывод программы-1 направляется на вход программы-2

цепочка: Program1 | Program2 | Program3

