A laptop screen with a dark frame. The background is a vibrant image of the Milky Way galaxy. A terminal window is open, showing a command prompt and its output. The terminal window has a title bar with three colored circles (red, yellow, green) and the text 'valentin@kirilov: ~'.

valentin@kirilov: ~

```
valentin@kirilov:~$ cat $(ls | grep pipes.*)
```

```
Pipes in Linux
```

```
Valentin Kirilov
```

```
FMI, Sofia University
```

```
Software Engineering, Operating Systems Course
```

```
valentin@kirilov:~$
```

# Pipe в Linux

Валентин Кирилов

# Какво ще разгледаме

- Какво е Pipe?
- Приложения в света около нас
- Pipe в Linux
- Реализация на Pipe







# Дефиниция

A ***pipe*** is a form of *redirection* that is used in **Linux** and other **Unix-like operating systems** to send the output of one **program** to another program for further processing.

Източник: <http://www.linfo.org/pipes.html>

# Пренасочване на поток

- Какво е поток?
- Какво е пренасочване?
- Повече информация: [www.linfo.org/pipes.html](http://www.linfo.org/pipes.html)
- Пример:

```
cat file_to_redirect.txt > redirection_output.txt  
./run_me < data.in  
./calculate_fib < in.txt > out.txt
```

# Пример 1

```
valentin@kirilov: ~/Documents/SU-FMI
valentin@kirilov:~/Documents/SU-FMI$ ls
FMI-2014-2015      Mini-Mathematica      project_SI_2_3_61701.pdf
group3.fn61701.kp.zip  npm-debug.log          Snake
ls_ouput.txt       project_SI_2_3_61701.docx
valentin@kirilov:~/Documents/SU-FMI$ ls -l > ls_ouput.txt
valentin@kirilov:~/Documents/SU-FMI$ ls -l | grep ls*
-rw-rw-r--  1 valentin valentin    585 Mar 18 20:55 ls_ouput.txt
valentin@kirilov:~/Documents/SU-FMI$
```



# Пример 2

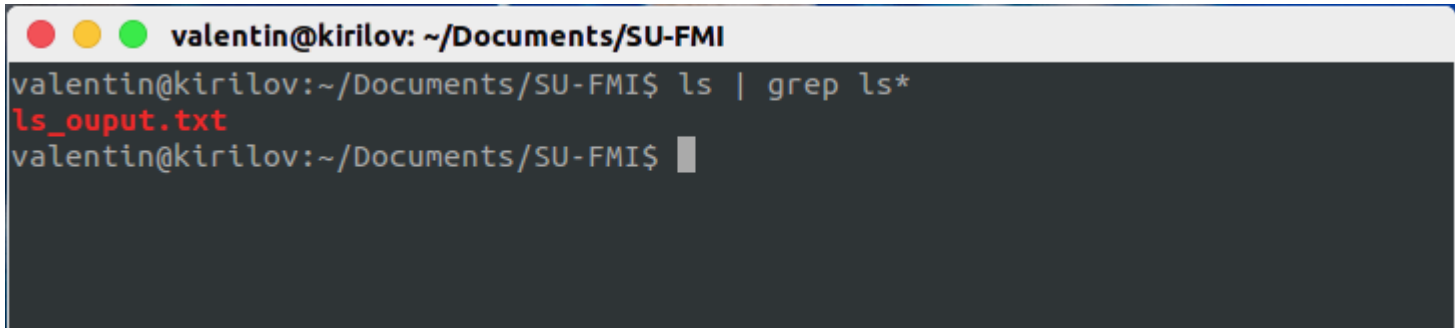
```
valentin@kirilov: ~/Documents/SU-FMI
valentin@kirilov:~/Documents/SU-FMI$ wc < ls_ouput.txt
  9  74 585
valentin@kirilov:~/Documents/SU-FMI$ wc < ls_ouput.txt > wc_ouput.txt
valentin@kirilov:~/Documents/SU-FMI$ cat wc_ouput.txt
  9  74 585
valentin@kirilov:~/Documents/SU-FMI$
```

# Какво е Pipe?

Пренасочването на изходните данни от даден процес към входа на друг.

- `ls | wc`
- `ls -l | grep $USER`
- `cat /etc/hosts | grep ip6 | wc`

# Пример 1



```
valentin@kirilov: ~/Documents/SU-FMI
valentin@kirilov:~/Documents/SU-FMI$ ls | grep ls*
ls_output.txt
valentin@kirilov:~/Documents/SU-FMI$
```

A terminal window with a title bar containing three colored circles (red, yellow, green) and the text "valentin@kirilov: ~/Documents/SU-FMI". The terminal shows a command "ls | grep ls\*" being executed, which results in the output "ls\_output.txt". The prompt "valentin@kirilov:~/Documents/SU-FMI\$" is shown before and after the command.

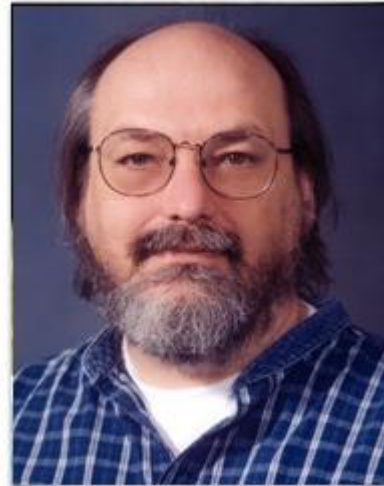
# Пример 2

```
valentin@kirilov: ~/Documents/SU-FMI
valentin@kirilov:~/Documents/SU-FMI$ ps
  PID TTY          TIME CMD
 7431 pts/0        00:00:00 bash
 7803 pts/0        00:00:00 ps
valentin@kirilov:~/Documents/SU-FMI$ ps | grep bash
 7431 pts/0        00:00:00 bash
valentin@kirilov:~/Documents/SU-FMI$
```

# Малко История



Malcolm Douglas  
**McIlroy**



Ken Thompson

# Еlegantни ф-сти на UNIX

- Каналите (pipelines)
- йерархичните файлови системи  
(*hierarchical file system*)
- регулярните изрази (regular expressions)



# Малко история (2)

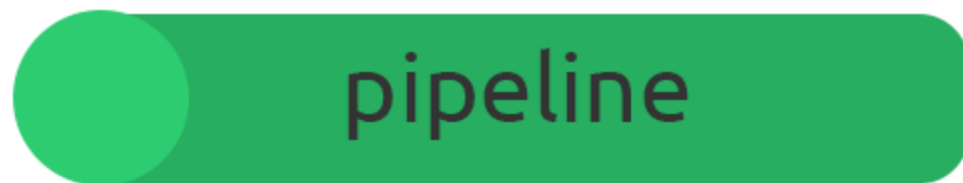
- 1973г. - включени в UNIX
- основна концепция за модуларност (modularity)

# Реализация

В примери

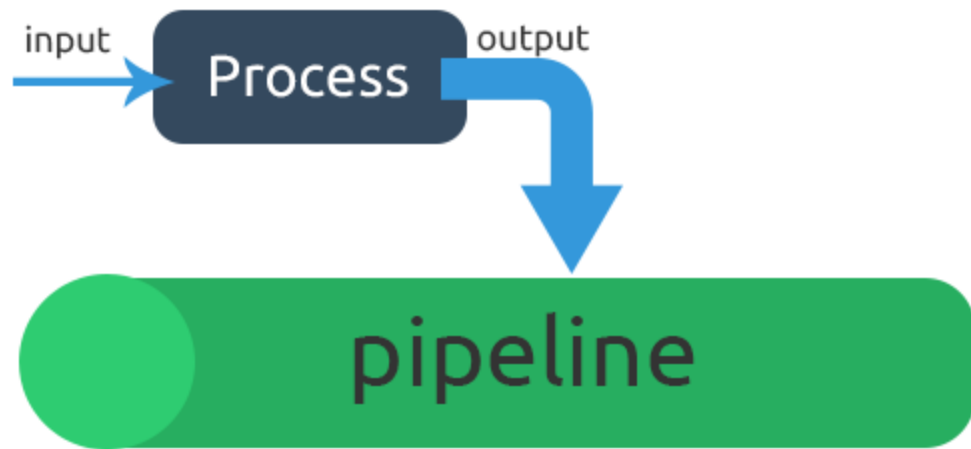


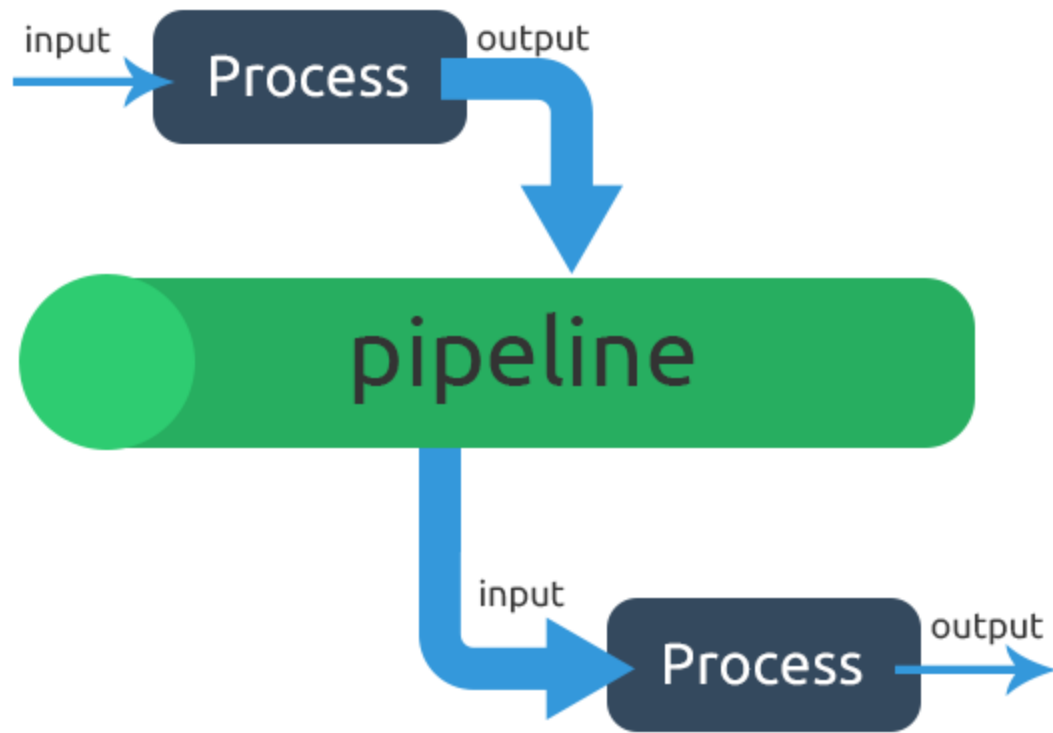












# Реализация

В код

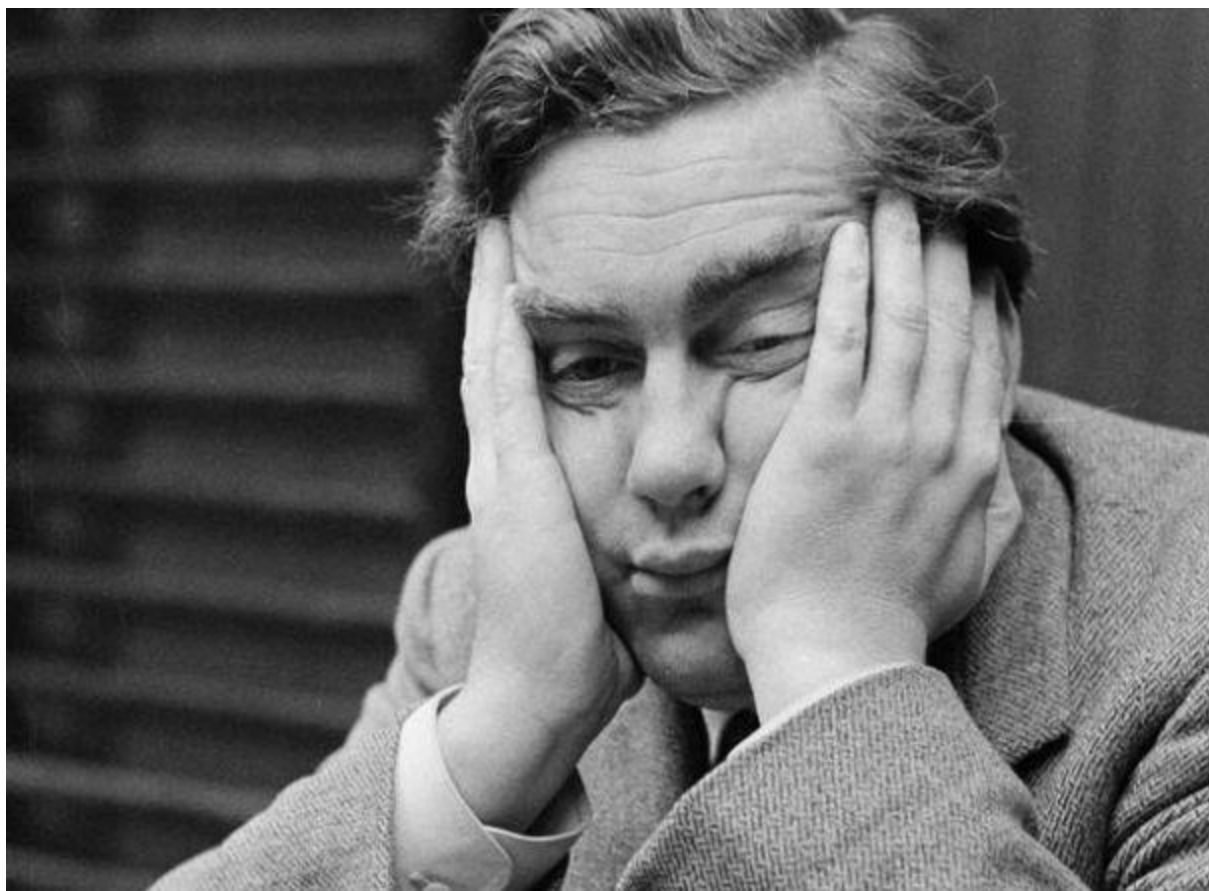
```
#define R 0 // дефинираме константа указваща пишещия процес
#define W 1 // дефинираме константа указваща четящия процес

int my_pipe[2]; // създаваме масив, с двата процеса
pipe(my_pipe); // създаваме pipe между двата процеса

pid_t pid = fork(); // създаваме си дъщерен процес, копие на текущия
if (pid == 0){ // блок за дъщерния процес
    // child
    close(my_pipe[W]); // затваряме станд. вход на процеса
    char buff[6]; // създаваме си буфер, в който да четем
    read(my_pipe[R], buff, 6 * sizeof(char)); // четем от изх.на първия процес
    close(my_pipe[R]); // затваряме изхода на първия процес

    write(STDOUT_FILENO, buff, 6 * sizeof(char)); // изписваме резултата
}
else if (pid > 0) { // блок за главния процес
    // parent
    close(my_pipe[R]); // затваряме станд. изхода
    char hello[] = "Hello\n"; // създаваме стринг
    write(my_pipe[W], hello, 6 * sizeof(char)); // пишем низа към входа на първия проц.
    close(my_pipe[W]); // затваряме входа на първия процес
}
else {
    perror("fork()"); // настъпила е грешка
}
```









# Приложения

- филтриране на данни
- манипулация на данни
- pipe в MongoDB
- задачи по математика



**Въпроси?**

# Благодаря!

29.04.2015

ФМИ, София