C2110 UNIX and programming

4th lesson

Petr Kulhánek, Jakub Štěpán

kulhanek@chemi.muni.cz

National Centre for Biomolecular Research, Faculty of Science
Masaryk University, Kotlářská 2, CZ-61137 Brno

european social fund in the czech republic
EUROPEAN UNION
MINISTRY OF EDUCATION, YOUTH AND SPORTS
OP Education for Competitivness

INVESTMENTS IN EDUCATION DEVELOPMENT

CZ.1.07/2.2.00/15.0233

Contents

- **File system II**
  - **Special symbols, quotas, access permissions, disk devices, USB disks, file search**
- **Processes I**
  - **Process, multitasking, basic commands,**
  - **Redirection, pipes**

File system II

- Special symbols
- Quotas
- Access permissions
- Disk devices
  - USB disks
- File search

Special symbols

Special symbols in file and directory names:

*****	- any number of any characters (except hidden files)
?	- one symbol
[]	- one symbol from listed values, example: [ajk], [a,j,k], [a-j]

Expansion of special symbols is done by shell **before submission** of a command.
Expansion can be **prevented** by **quotation** marks or by **slash** symbol before special symbol.

Examples

```
$ cp *.pdf Documents/
```

copy all pdf documents from actual directory to subdirectory Documents

```
$ rm *
```

remove all files in current working directory (except directories)

```
$ mv A? Tmp/
```

move files with name beginning with "A" and with total 2 characters long name to directory "Tmp"

Special symbols

Expansion is done only when there is at least one file fulfilling the given request:

Examples:

```
$ cd
$ echo D*
Desktop Documents Downloads
$ echo A*
A*
$ echo "D*"
D*
$ echo D\*
D*
```

Quotas

There is disk usage quota for your home directories on WOLF cluster on **wolf.wolf.inet:/export/home/**. Current quota setup and disk usage may be printed by command **quota**:

```
[kulhanek@wolf ~]$ quota -vs
Disk quotas for user kulhanek (uid 18773):
     Filesystem  blocks   quota   limit   grace   files   quota   limit
wolf.wolf.inet:/export/home/
                  1550M   1954M   2051M           20453       0       0
```

Current usage

Hard limit, cannot be exceeded.

Limit that can be exceeded temporarily (1 week)

Quota exceeding may lead to **login inability** to graphic environment. In that case, login to text terminal (Ctrl+Alt+F1) and move some data to another disk device (for example to directory /scratch/your_login or remove some data).

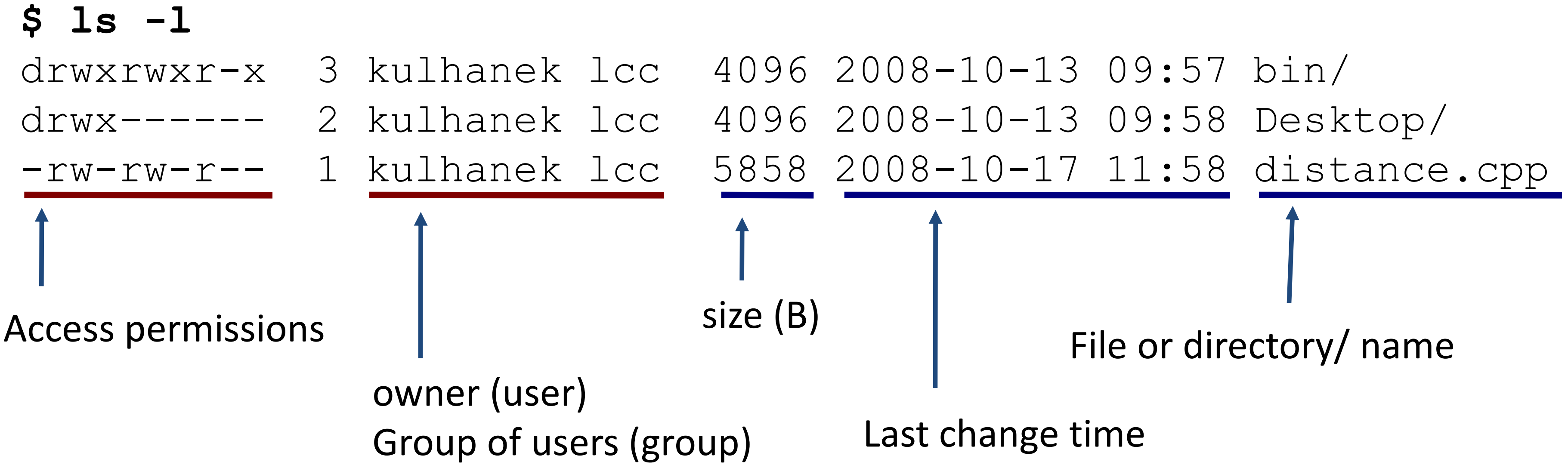
Access permissions

Access permissions determine what operations user can do with file or directory in file system.

Access permissions:

r	read file	list directory contents
w	change file	change directory contents
x	run file	enter directory

Each file and directory has one owner and group. Access permissions are divided into 3 sections: user – u, group – g, other users – o.

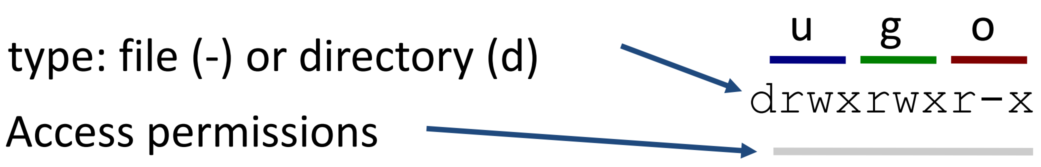


```
$ ls -l
drwxrwxr-x   3 kulhanek lcc   4096 2008-10-13 09:57 bin/
drwx------   2 kulhanek lcc   4096 2008-10-13 09:58 Desktop/
-rw-rw-r--   1 kulhanek lcc   5858 2008-10-17 11:58 distance.cpp
```

Access permissions change

Owner or super-user can change file or directory access permissions by command **chmod.**

```
$ chmod permissions file1 [file2 ...]
```



Access permissions:

r	read file	list directory contents
w	change file	change directory contents
x	run file	enter directory

Permissions sections:

u	owner (user)
g	user group (group)
o	others
a	all users, applied to u,g,o

Example:

```
$ chmod u+x,g-w file
```

Add (+) permission to run to owner and remove (-) permission to write to group.

Owner change, group change

File owner can be changed only by super-user by command **chown.**

Group of files and directories may be changed by owner or super-user by command **chgrp**. Owner may use only groups that he is member of (obtain list by command **id**).

```
$ chgrp group_name file1 [file2 ...]
```

```
[kulhanek@wolf01 ~]$ id
uid=18773(kulhanek) gid=2001(lcc) groups=2001(lcc),2027(kulhanek),2030(compchem)
```

```
[kulhanek@wolf01 ~]$ ls -ld Documents/
drwxr-xr-x 9 kulhanek lcc 4096 Feb 16  2012 Documents/

[kulhanek@wolf01 ~]$ chgrp compchem Documents/

[kulhanek@wolf01 ~]$ ls -ld Documents/
drwxr-xr-x 9 kulhanek compchem 4096 Feb 16  2012 Documents/
```

Group change

Disk devices

Overview of file system usage, disk devices and mount points is printed by command **df**.

```
[kulhanek@wolf01 ~]$ df -Th
Filesystem                           Type      Size  Used Avail Use% Mounted on
/dev/mapper/server1-root             ext3       20G  5.9G   13G  32% /
udev                                 devtmpfs  3.9G  4.0K  3.9G   1% /dev
tmpfs                                tmpfs     1.6G  444K  1.6G   1% /run
none                                 tmpfs     5.0M     0  5.0M   0% /run/lock
none                                 tmpfs     3.9G   12K  3.9G   1% /run/shm
/dev/sda1                            ext3      168M   36M  124M  23% /boot
/dev/mapper/server1-scratch          ext3       20G  1.9G   17G  11% /scratch
/dev/mapper/server1-vbox             ext3       20G  5.6G   14G  30% /win
wolf.wolf.inet:/export/software/ncbr nfs        93G   60G   29G  68% /software/ncbr
wolf.wolf.inet:/export/home          nfs       280G  164G  102G  62% /auto/home
```

device **File system type** **Mount point**

File system types:

ext3,ext4 – third / fourth extended file system (native Linux file system)
nfs – network file system
vfat – Virtual File Allocation Table (used in MS Windows)
ntfs - New technology File System (Microsoft file system)

USB disks

USB disks are connected (mounted) automatically in graphical environment to destination **/media**.

```
[kulhanek@wolf01 ~]$ df -Th
Filesystem                          Type      Size  Used Avail Use% Mounted on
................................................................................
wolf.wolf.inet:/export/home         nfs       280G  164G  102G  62% /auto/home
/dev/sdg1                           vfat      962M  841M  122M  88% /media/B19A-1CA2
```

Disk may be unmounted in graphical environment or tin command line by command **umount**. Command needs as an argument mount point of device.

```
[kulhanek@wolf01 ~]$ umount /media/B19A-1CA2
```

Disk can be unmounted only if there is no process opened disk path (mount point) or using the disk data. Overview of processes using directory contents (mount point) can be printed by command **lsof** (or fuser).

```
[kulhanek@wolf01 ~]$ lsof /media/B19A-1CA2/
COMMAND PID      USER    FD    TYPE DEVICE SIZE/OFF NODE NAME
bash    31521 kulhanek  cwd    DIR    8,97     4096  518 /media/B19A-1CA2/GoslarFinal
bash    31893 kulhanek  cwd    DIR    8,97     4096  518 /media/B19A-1CA2/GoslarFinal
vi      32011 kulhanek  cwd    DIR    8,97     4096  518 /media/B19A-1CA2/GoslarFinal
vi      32011 kulhanek    4u   REG    8,97    12288  535 /media/B19A-1CA2/GoslarFinal/.README.swp
```

File search

To find files one can use command **find**.

If not used – current working directory is used.

```
$ find [where] what
```

Search is recursive (default)

Search query (**what**) is given from parts connected by logical operators.

Most common queries:

-name *pattern*	finds all files that have name *pattern* *pattern* may contain special symbols: *,?,[] **(when using special symbols, we use *pattern* with quotation marks – in this case we want command to expand special symbols, not shell)**
-type *c*	find files of type ***c*** (file, directory, etc., get type list in find man page)

Logic operators:

-and	left and right queries are fulfilled together
-or	left **or** right query is fulfilled

File search, examples

```
$ find /home/ -name '*.txt'
```

in directory /home/ find all files with extension .txt

```
$ find ~kulhanek -name '*.txt' -or -name '*.hpp'
```

in directory /home/kulhanek find all files with extension .txt or .hpp

```
$ find -name 'D*' -and -type d
```

in current working directory find all subdirectories with name beginning with character D

Command overview

File system:

ls	print directory contents
pwd	print current working directory
cd	change current working directory
mkdir	create directory
rmdir	remove empty directory
cp	copy file or directory
mv	move file or directory
rm	remove file or directory
find	find file or directory
du	print size of file or directory
stat	print info of file or directory
df	print info of mounted disk devices
quota	print info of quotas on file system
scp	remote secure copy over network

Exercise

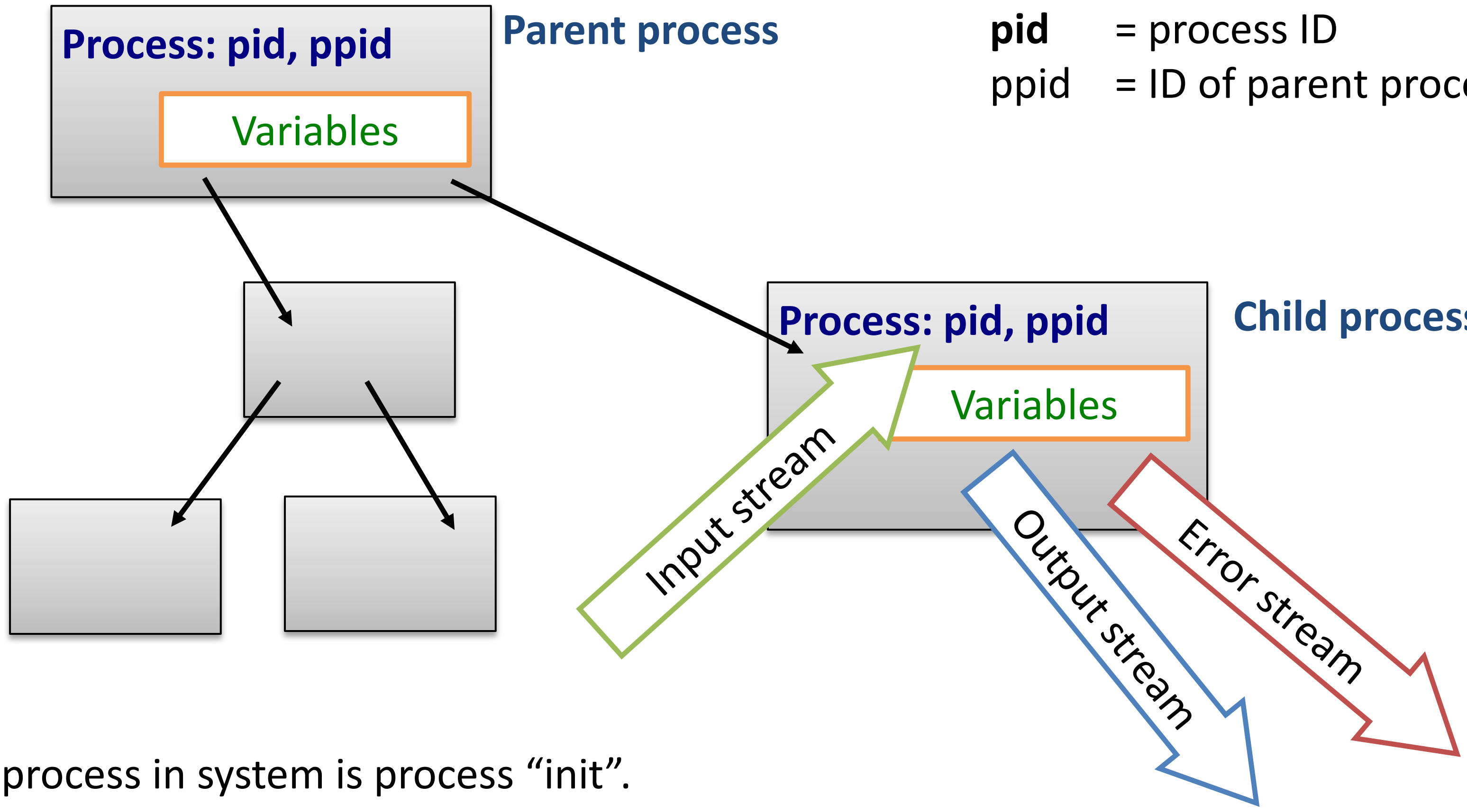
1. In home directory create subdirectory **Data**
2. Copy contents of **/home/kulhanek/Data/** with subdirectories to **Data**
3. Find all files with extension **.cpp**, that are in directory **Data** (print to screen)
4. In directory **/scratch/your_login** create directory **Headers**
5. To **Headers** directory, copy all files from /**home/kulhanek/Data/dev/src** , with extension **.h**
6. Move all files with extension **.cpp** from directory /**home/your_login/Data/dev/src** , to **Headers**
7. What is size of file /**home/kulhanek/Data/dev/src/GraphicsSetup.cpp** in B and kB
8. Make sure that your data are within limits of quota in your home directory.
9. Remove all files beginning with **Graphics** and with extension **.h** in directory **Headers**

Processes I

- Process
- Multitasking
- Basic commands
- Redirection
- Pipes

Procesy

Process is running **program**. In any moment on one physical processor, only one process can be running. Operating system then ensures fast switches among running processes so that they seem to be running simultaneous run (**multitasking**).



pid	= process ID
ppid	= ID of parent process

- First process in system is process "init".
- Each command submitted to command line shellu is process.

List of processes

Process list can be printed by commands:

top prints processes by their CPU time consumption – periodic refresh (finish by key q)

ps print processes running in terminal (options can print all processes and various information)
(`ps -u user_name`)

pstree process tree print

```
$ ps
  PID TTY          TIME CMD
 8763 pts/5    00:00:00 bash
 8852 pts/5    00:00:00 gimp
 8857 pts/5    00:00:00 ps
```

Running command name

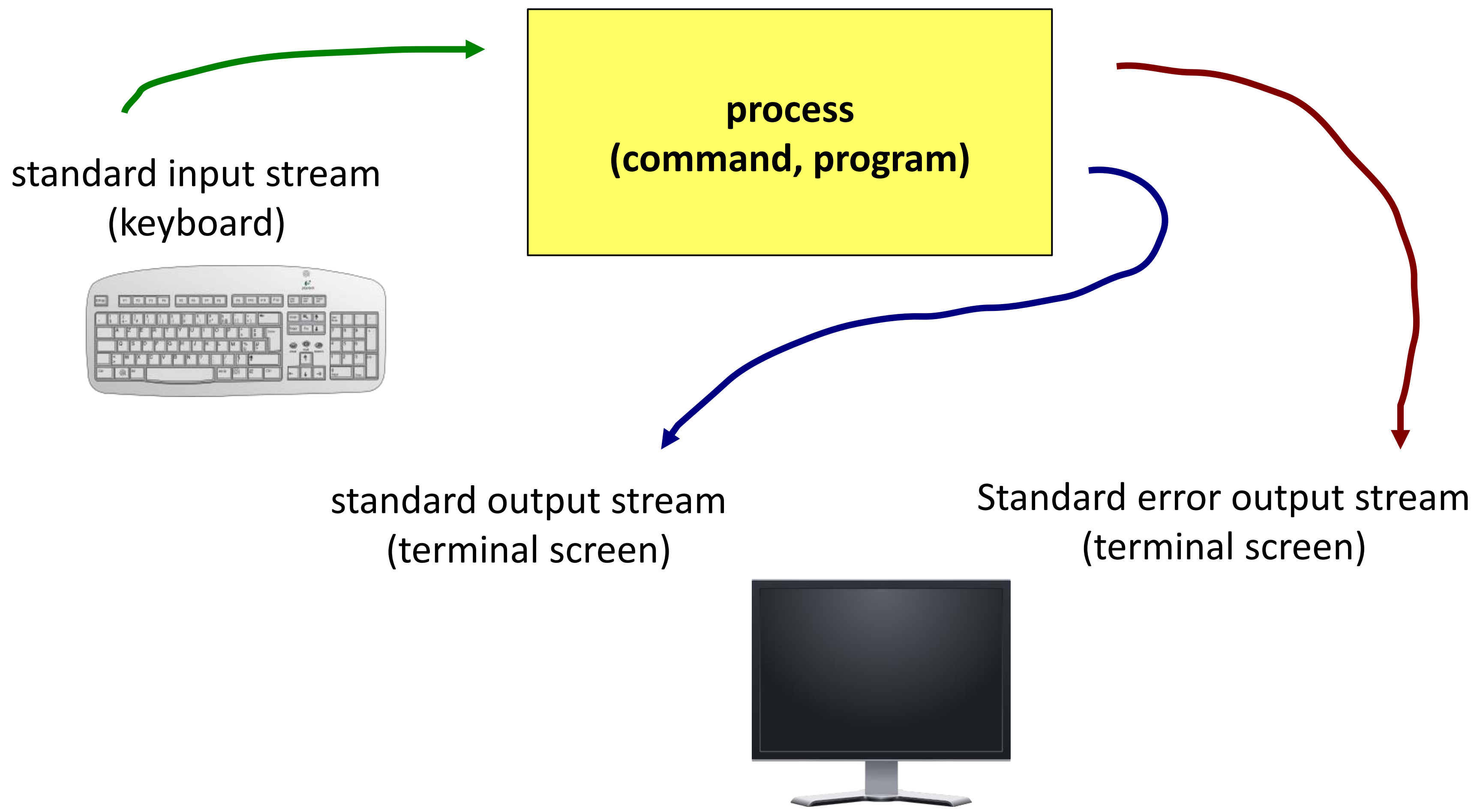
Consumed CPU time

Process number

Terminal, where process is running

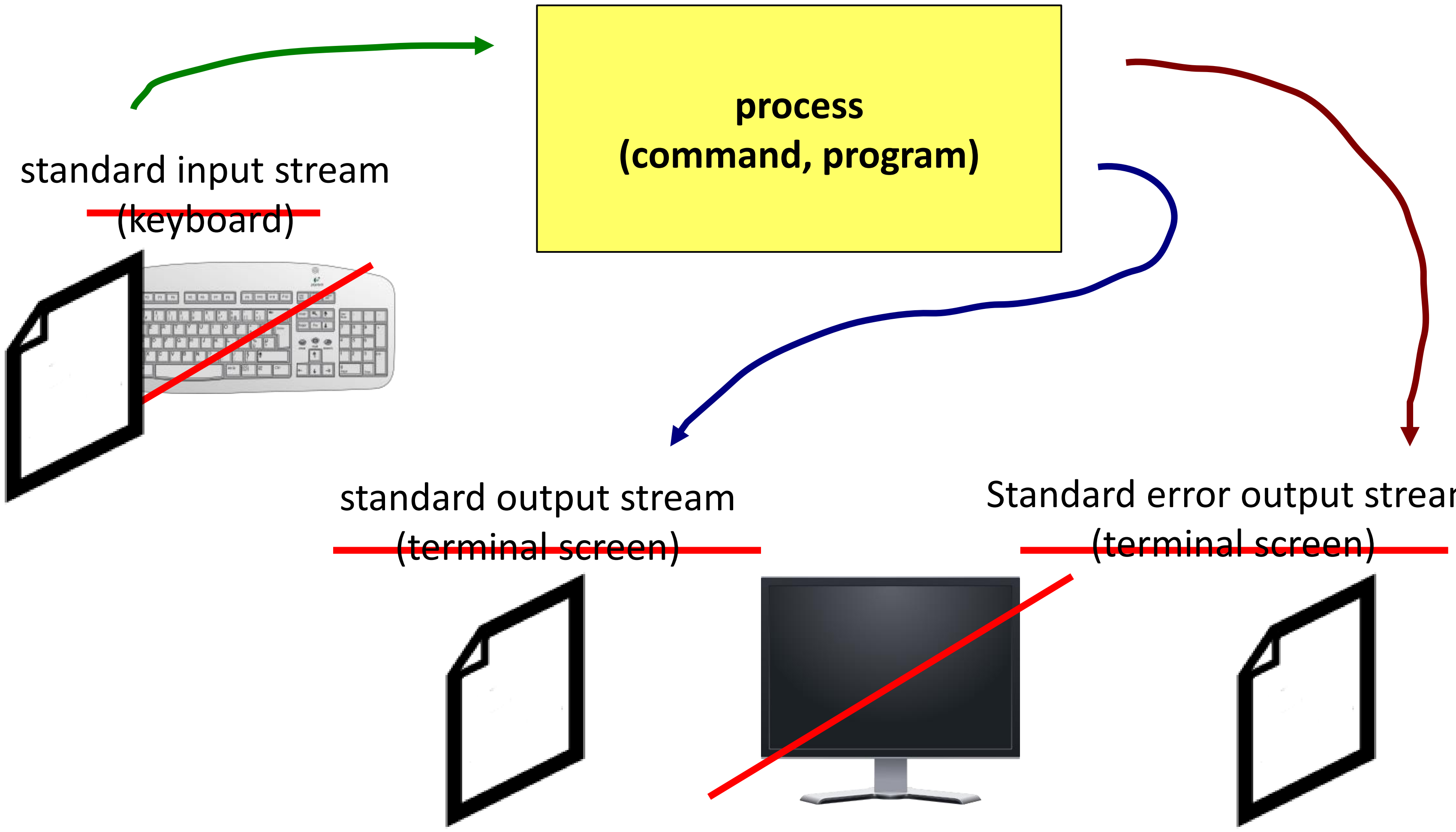
Standard streams

Input – output streams serves process for **communication** purposes with environment. Any process has **three standard streams**:



Redirection

Input – output streams may be redirected in such way, that **files** are used instead keyboard or screen.



Input redirection

Standard input redirection of program called my_command from file **input.txt**.

```
$ my_command < input.txt
```

Standard input redirection of program my_command from script file.

```
.......
./my_command << EOF
prvni radka textu
druha radka textu
treti radka textu
EOF
......
```

Mark that denotes input (user choice form)

text, that creates input

Input end, *mark* ***must not*** *be surrounded by spaces*

This redirection is especially appropriate for usage in scripts, it work also in command line. Advantage is direct variable expansion in text.

Output redirection

Standard output redirection of program my_command to file **output.txt**. (File is created, if exists, its contents are **replaced**.)

```
$ my_command > output.txt
```

Standard output redirection of program my_command to file **output.txt**. (File is created, if exists, data are **added** to its end).

```
$ my_command >> output.txt
```

Similar rules apply for **error output**, following operators are used:

```
$ my_command 2> errors.txt
$ my_command 2>> errors.txt
```

Standard streams connection

Standard output **and** standard error output of program my_command may be redirected to single file **output.txt**.

```
$ my_command &> output.txt
```

Cannot be applied to operator >>.

~~`$ my_command &>> output.txt`~~ NO.

Solution: First **redirect** standard output and then **connect** standard and error streams to one.

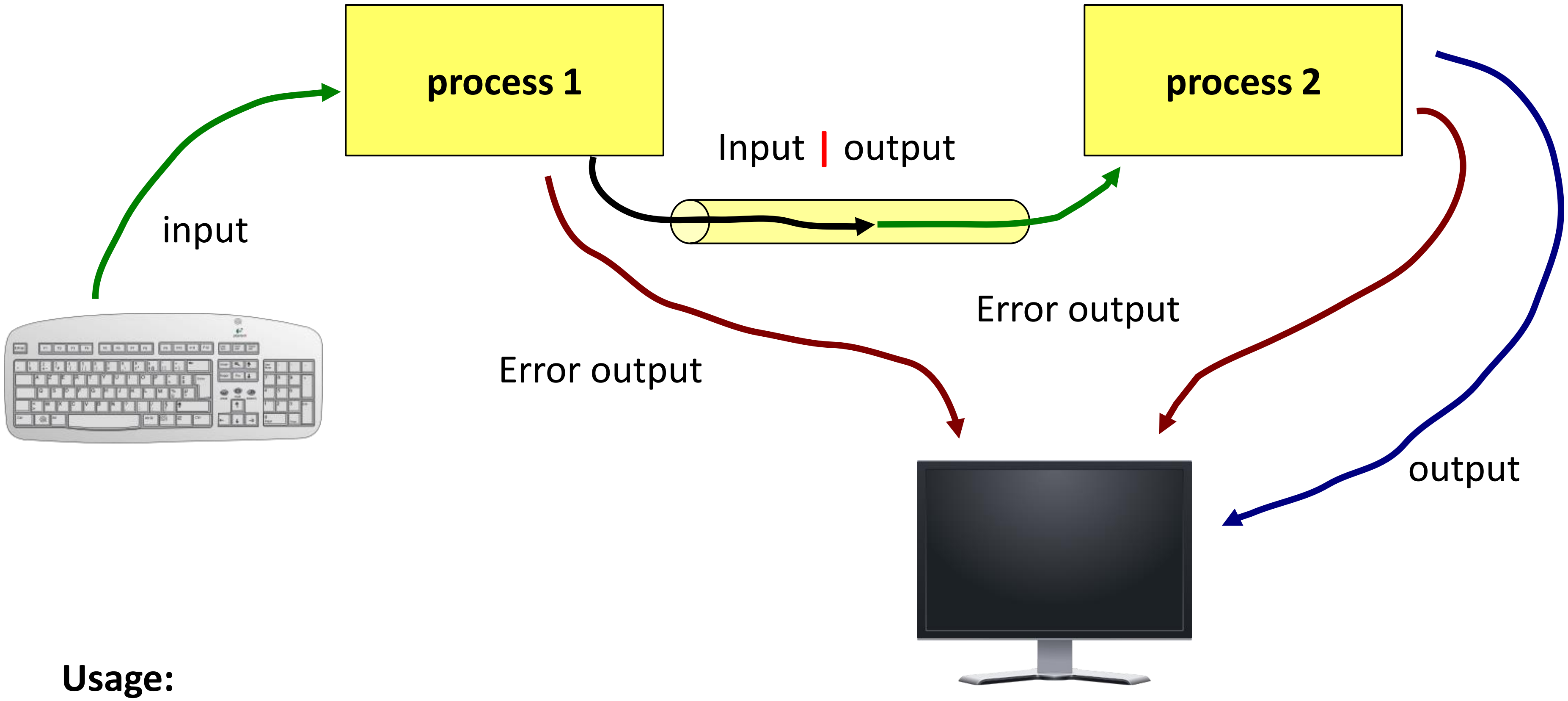
```
$ my_command >> output.txt 2>&1
```

Order is important.

~~`$ my_command 2>&1 >> output.txt`~~ NO.

Pipes

Pipes serves to connect standard output of one process with standard input of another process.

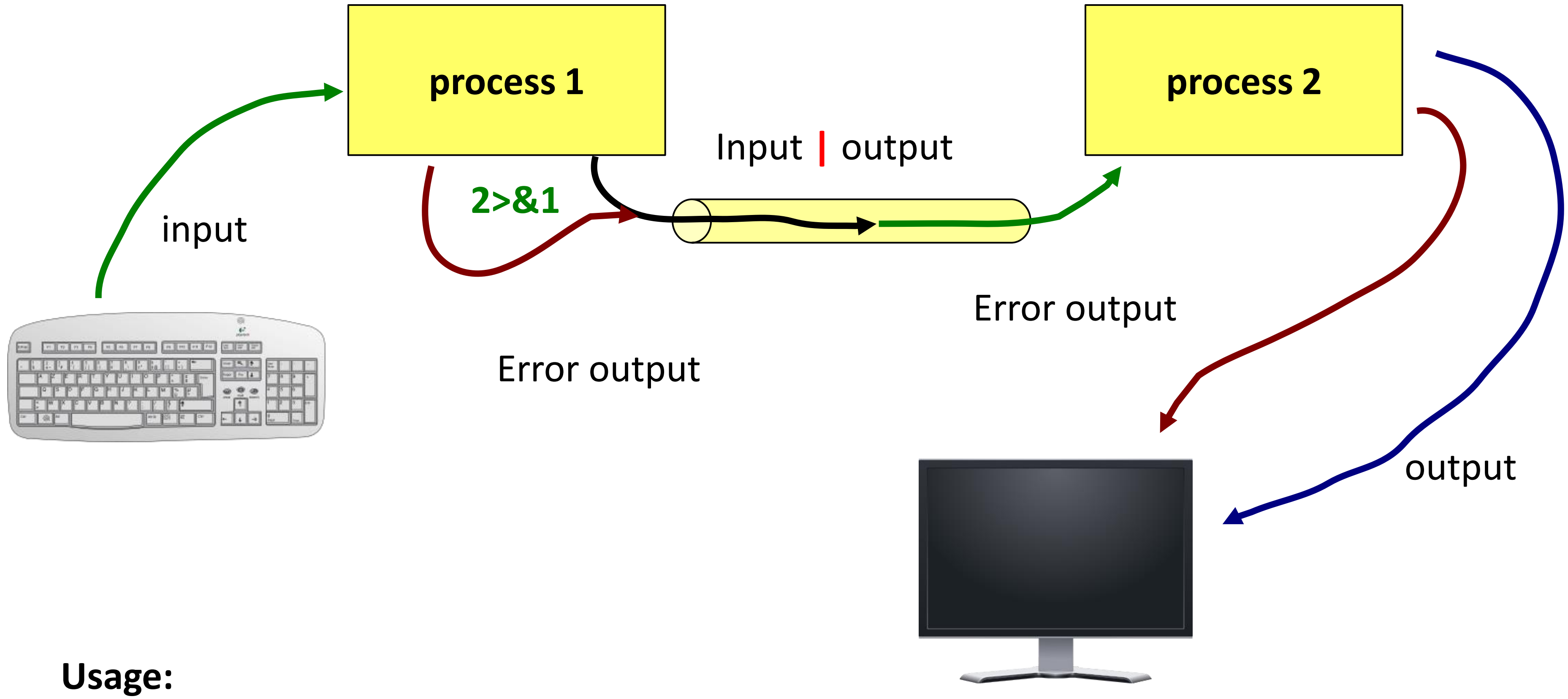


Usage:

```
$ command_1 | command_2
```

Pipes and error output

Transfer of standard error output over pipe is possible by connecting output streams to one.



Usage:

```
$ command_1 2>&1 | command_2
```

Examples

cat	prints file contents (one after another)
paste	prints file contents (next to each other)
wc	prints word, character and lines count of file
head	prints file from beginning
tail	prints file from end
xargs	uses data from standard input (passed by pipe) as arguments of command, not for standard input

Examples:

```
$ cat file1.txt file2.txt
```
prints contents of files file1.txt a file2.txt to screen, one after another

```
$ paste file1.txt file2.txt
```
prints contents of files file1.txt and file2.txt to screen next to each other

```
$ wc file.txt
```
prints word, character and lines count of file file.txt

```
$ head -15 file.txt
```
prints first 15 lines of file file.txt

```
$ tail -6 file.txt
```
prints last 6 lines of file file.txt

Exercise

1. **Processes, access permissions**
   a) Create directory **Processes** in your home directory
   b) Copy command (file of program) **/bin/ls** to directory **Processes** and rename it to **myls**
   c) Run program **myls**
   d) What access permissions does file **myls** have?
   e) Remove run permissions of file **myls** for all groups. What happens if you try to run command again?

Exercise

2. Pipes, redirection

a) Find all files with extension **.f90**, in directory **/home/kulhanek/Data/dev/src/** , put list into file **~/Procesy/list.txt**

b) How many lines does file **list.txt** have?

c) Print first two lines of file **list.txt** to screen and then also to file **two_lines.txt**

d) Print only third line of file **list.txt**

e) Find all files beginning with characters **cpu** in directory **/proc**. Remove all error messages from output by redirecting them to **/dev/null**

f) Try following command constructions in home directory. Why they differ? What does command **xargs** exactly do?

- echo Documents | xargs ls
- echo Documents | ls