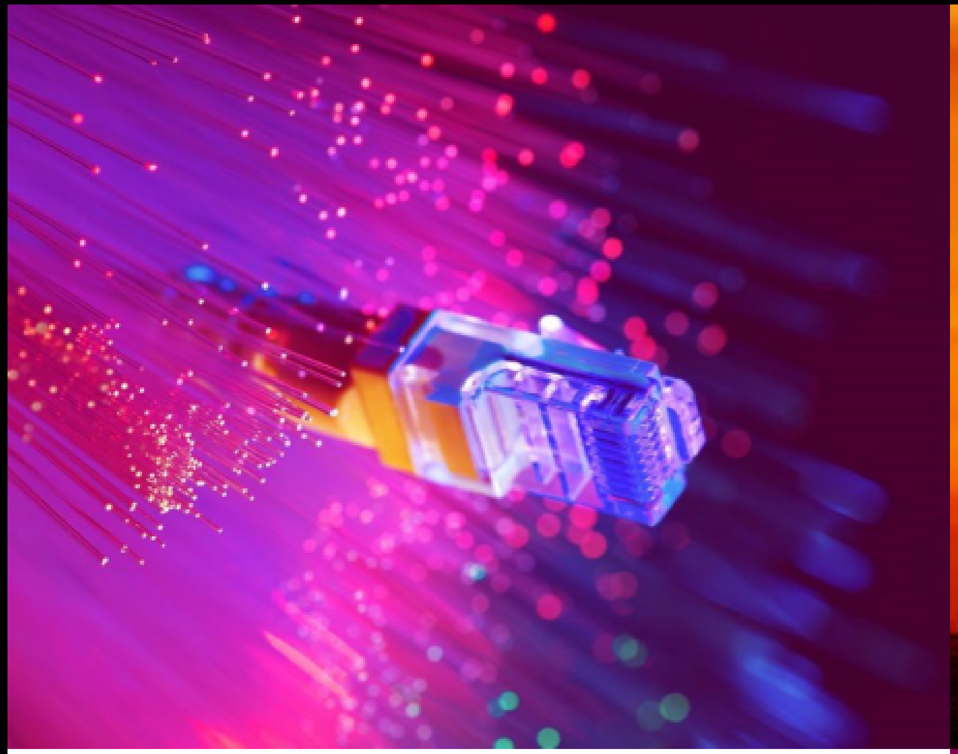
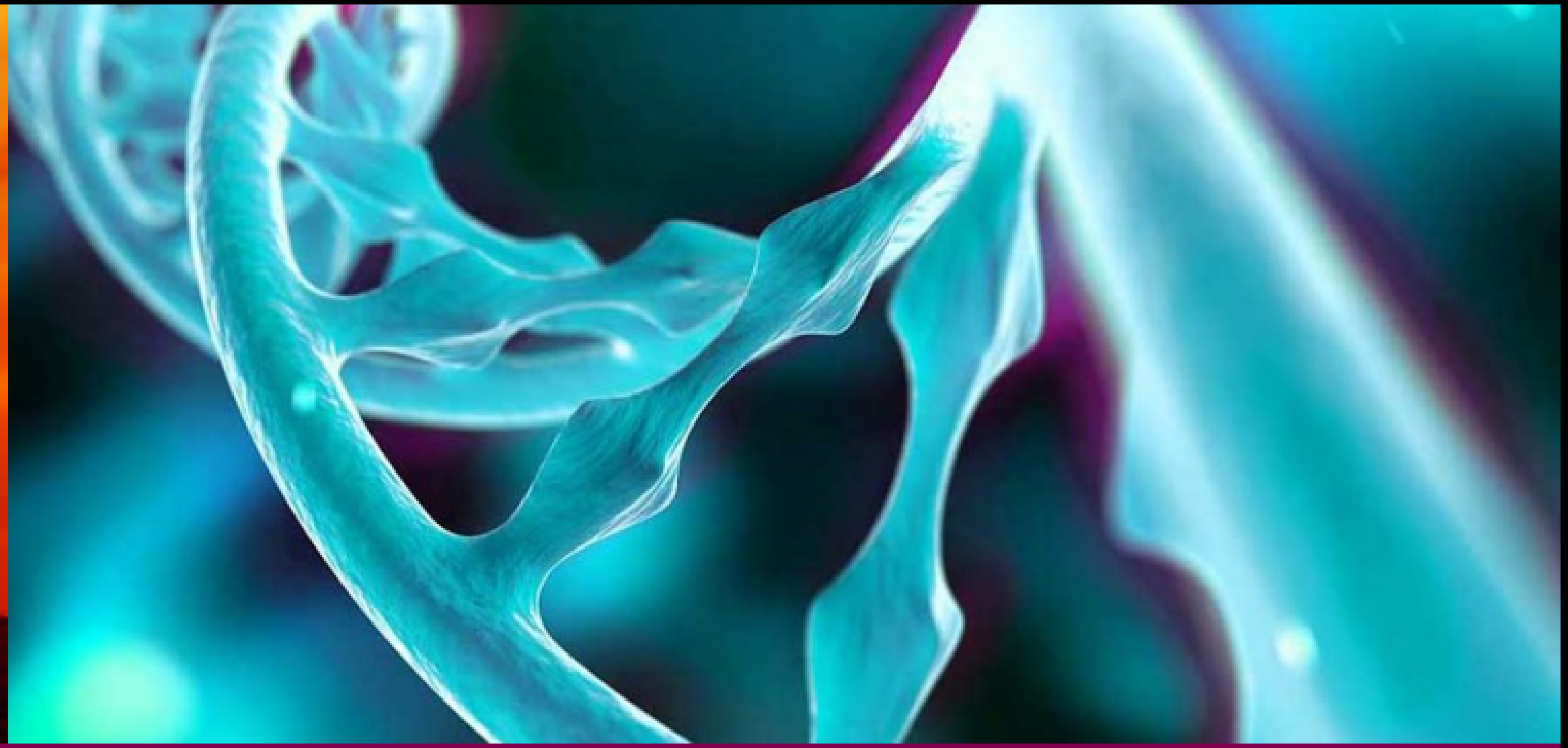




Pipes and redirecting

CSC Training, 2019-12

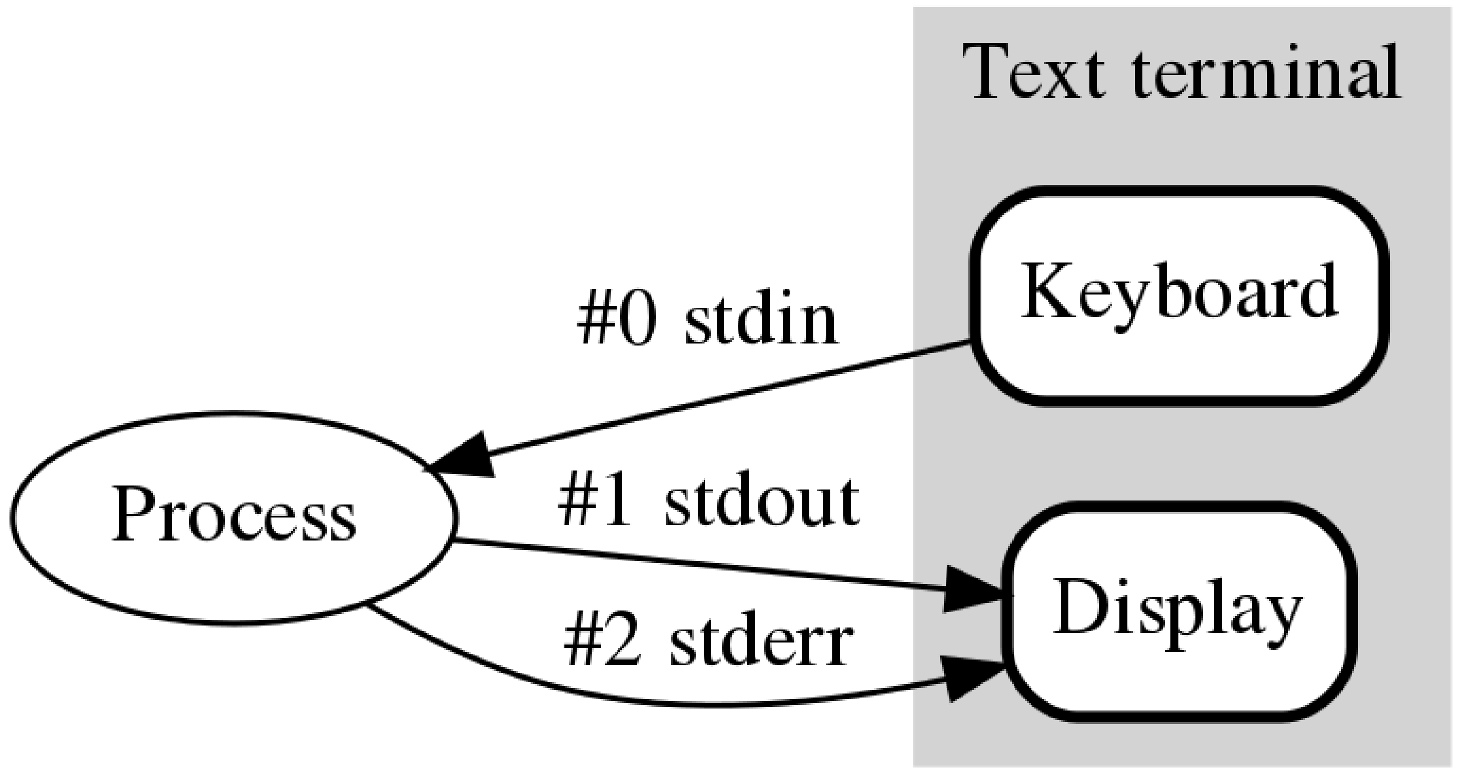
CSC – Finnish expertise in ICT for research, education and public administration

File descriptors

- A file descriptor *is an abstract indicator used to access a file or other input/output resource, such as a pipe or network connection.* (Wikipedia)
- A file descriptor is a non-negative integer number, which is used as a reference pointer to a file or other input/output resource.
- Three standard file descriptors:
  - **Standard input**, stdin
  - **Standard output**, stdout
  - **Standard error**, stderr

File descriptors (cntd)

stream	Descriptor	default direction
stdin	0	terminal input
stdout	1	terminal output
stderr	2	terminal output



Redirecting

- Before a command is executed, its input and output may be redirected using redirection operators |, <, > and/or >>.
    - The pipeline is a simple redirection where output (stdout) of each command in the pipeline is connected via a pipe to the input (stdin) of the next command.

```
$ ls | wc -l
```

- The redirection operators may precede or appear anywhere within a simple command or may follow a command.
- Redirections are processed in the order they appear, from left to right.

Redirecting output

- The general format for redirecting output is: `[n]>file` or `[n]>&[m]`
  - Redirection of output causes the *file* to be opened for writing on file descriptor *n* (default on standard output). *m* is another file descriptor.

```
$ grep -r "global" /etc   >output    # redirects default stdout
$ grep -r "global" /etc  2>error     # redirects stderr
```

- Multiple redirections with descriptor duplication:

```
$ grep -r "global" /etc >output 2>error   # redirects stderr and stout to separate files
$ grep -r "global" /etc >output 2>&1      # redirects stderr to stdout to file
```

- Redirecting to a pipe:

```
$ grep -r "global" /etc  2>&1 | less
```

Appending redirected output

⚠ Unless the shell option `noclobber` has been set redirecting to a file always overwrites an existing file ⚠

- To prevent accidental overwriting:

```
$ set -o noclobber
```

- If you need to **append** to a *file* instead of overwriting it, replace the > with >> operator:

```
$ ls >>output
$ grep -r "global" /etc  >>output 2>&1
```

Some special files

- Unix has three special, so-called *pseudo device files*, which are commonly being used with redirections:
  - `/dev/null` discards all data written to it but reports that the write operation succeeded. Trying to read from it provides nothing.

```
$ grep -r "global" /etc 2>/dev/null
```

  - `/dev/zero` provides an endless stream of zeros (bytes, not character).
  - `/dev/random` provides an endless stream of random bytes.

```
$ cat /dev/random >gibberish.txt    # end with CTRL+C
$ ls -l gibberish.txt
```

Redirecting input

- The general format for redirecting input is: `[n]<file`
- Redirection of input causes the file to be opened for reading on file descriptor `n` (or by default the standard input).

```
$ tr a-z A-Z <adg.txt
$ <adg.txt >output tr a-z A-Z
```

Here-documents

- A here-document is an input redirection using source data specified directly at the command line until a line containing only a certain keyword is seen.
- The format of here-documents is:

```
command <<[-]keyword
    contents of
    here-document
keyword
```

- All of the lines read up to that point are then used as the standard input for a command.
- See examples next slide.

Here-documents (cntd)

- Often used in scripts for printing out instructions:

```
$ cat <<EndOfText
  A line of text.
  Second line of text.
EndOfText
```

- …but it can also be used to steer an interactive program:

```
$ gnuplot <<end
set terminal dumb
f(x)=1/x
plot f(x)
end
```

✋ In case your system lacks `gnuplot`, install by: `sudo apt-get install gnuplot`

Here-documents (still cntd)

- A variant of here documents, the format is: `command <<<word`
- The *word* undergoes brace expansion, tilde expansion, parameter and variable expansion, command substitution, arithmetic expansion, and quote removal. The result is supplied as a single string to the command on its standard input.

```
$ ls <<< $PWD
$ bc <<< 2^10
```

```
$ tr a-z A-Z <<< 'one two three'
$ echo 'one two three' | tr a-z A-Z
```

```
$ read a b c <<< $(echo 'one two three'); echo $c $b $a
```

Named pipes

- A named pipe (FIFO) works much like a regular pipe, but does have some noticeable differences:
    - Named pipes exist as a device special file in the file system.
    - Processes of different ancestry can share data through a named pipe.
    - When all I/O is done by sharing processes, the named pipe remains in the file system for later use.

Named pipes (cntd)

- A named pipe is created with `mkfifo` command:

```
$ mkfifo -m a=rw MYFIFO
```

- FIFO files can be quickly identified in a physical file system by the “p” indicator seen here in a long directory listing:

```
$ ls -lF MYFIFO
[…]
prw-rw-rw-  1 userid  group  0 Dec 5 13:16 MYFIFO|
```

✋ Notice the vertical bar (“pipe sign”) located directly after the file name.

Named pipes (cntd)

0. If you haven't done so, create a fifo as described on the previous slide

1. In your terminal type:

```
$ cat <<EOF >MYFIFO
  Hello, is there anybody in there?
  Not if you can see me.
  Is there anyone at all?
EOF
```

2. In *another* terminal type:

```
$ cat <MYFIFO
```

3. In order to remove a named pipe:

```
$ rm MYFIFO
```