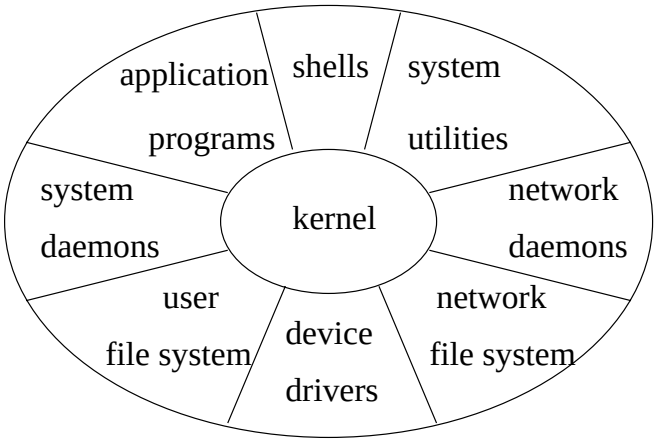
This lecture overview

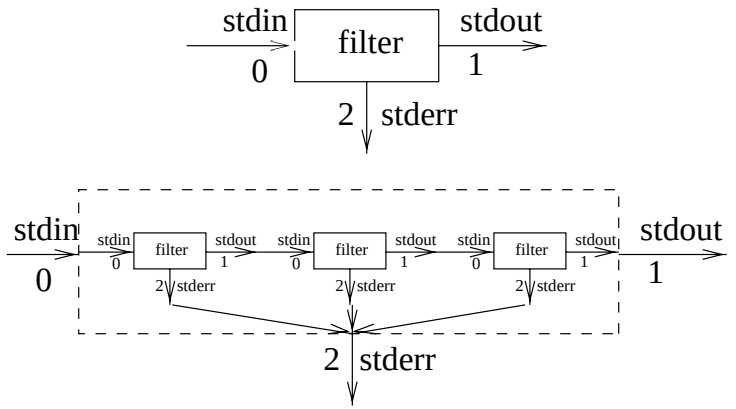
- UNIX structure.
- Basic commands.
- Programming in shell.

Structure of UNIX



ls – list directory contents
ls -l – list directory contents (long format)
pwd – print working directory
cd path – change directory
cat file – concatenate files and print on terminal
more file – display files one screenfull at a time
cp file1 file2 – copy *file1* into *file2*
cp file1 file2 ... directory – copy files into directory
mv file1 file2 – rename *file1* into *file2*
mv file1 file2 ... directory – move files into directory
mkdir directory – make directory
rm file1 file2 ... – remove (delete) files
rmdir directory – remove directory

Filters



ps -ef | grep jurek | wc -l

I/O redirection

- `>` – output redirection
- `2>` – error redirection
- `<` – input redirection
- `>>` – output redirection (append)
- `|` – redirect output of one program into the input of another one
- `2>&1` – redirect errors to standard output

Wildcards

- `*` – any string
- `?` – any character
- `[...]` – any character from the class
- `[A-Z]` – any upper case letter
- `[!...]` – any character not belonging to the class

Variables

PATH – command search path
HOME – users home directory
MAIL – users mailbox
SHELL – name of the shell
PS1 – command prompt
PS2 – command continuation prompt

Positional parameters

- `$0` – command name
- `$1` – first argument
- `$i` – i-th argument
- `$*` – all agruments exept `$0` (as single string)
- `$@` – all arguments exept `$0` (as separate strings)
- `$#` – number of positional parameters
- `$?` – exit status of last command
- `$$` – process number of this shell
- `$!` – process number of last background command

Parameter substitution

- `${name}` – substitute parameter value
- `${name:-word}` – if parameter is not set substitute word
- `${name:=word}` – if parameter is not set it to word
- `${name:?word}` – if parameter is not set print word and exit from the shell
- `${name:+word}` – if parameter is set substitute word

Quoting

Special characters: `; & ( ) | ^ < >` newline space tab

- `\c` – just character *c*
- `'string'` – quote all characters in the string
- `"string"` – quote all characters except `$`
- `` `command` `` – substitute the result of the command

Internal commands

- `break` – break from the loop
- `continue` – jump to the begining of the loop
- `cd` – change directory
- `echo` – write text
- `export` – make variables available for other programs
- `read` – read one line from the input
- `test` – test a condition
- `unset` – delete definition of the variable

Execution controll

- `for` *name* `[ in` *word* `... ] do` *list* `done` – Repeat *list* for *name* set to all listed words
- `case` *word* `in [` *pattern* `[ |` *pattern* `] ... )` *list* `;; ] ... esac` – Execute *list* if *word* matches *pattern*
- `if` *exp* `then` *list* `[ elif` *list* `then` *list* `] ... [ else` *list* `] fi` – Execute *list* according to expression
- `while` *exp* `do` *list* `done` – Execute *list* while expression is true
- `(`*list*`)` – Execute in subshell
- `{` *list*`;}` – Execute in this shell
- `name(){` *list*`;}` – Define procedure
- `#`*comment* – Just a comment

File tests

-r file – True if file exists and is readable
-w file – True if file exists and is writable
-x file – True if file exists and is executable
-f file – True if file exists and is regular file
-d file – True if file exists and is a directory
-h file – True if file exists and is symbolic link
-s file – True if file exists and has non zero size

Comparisons

-z string – True if the length of the string is zero
-n string – True if the length of the string is nonzero
s1 = s2 – True if strings s1 and s2 are identical
s1 != s2 – True if strings s1 adn s2 are not identical
n1 -eq n2 – True if integers n1 and n2 are equal
n1 -ne n2 – True if integers n1 and n2 are not equal
n1 -lt n2 – True if integer n1 is less than n2
n1 -le n2 – True if integer n1 is less or equal n2
n1 -gt n2 – True if integer n1 is greater than n2
n1 -ge n2 – True if integer n1 is greater or equal n2


```
# Waiting for response y or n
getyn()
{
  while echo "\n$* (y/n)? \c">&2
  do read yn rest
     case $yn in
     [yY]) return 0 ;;
     [nN]) return 1 ;;
        *) echo "Answer y or n" >&2 ;;
     esac
  done
}
getyn "Do you want to save the file"
if [ $? -eq 0 ]; then
....
```