



Politechnika  
Wrocławska

# Systemy operacyjne 2

Laboratorium nr 2

Podstawy skryptów powłoki Bash

Szymon Datko

[szymon.datko@pwr.edu.pl](mailto:szymon.datko@pwr.edu.pl)

Wydział Informatyki i Telekomunikacji,

Politechnika Wrocławska

semestr letni 2021/2022



HR EXCELLENCE IN RESEARCH

W skrócie – trzeba wiedzieć jakich programów użyć

# LINUX TERMINAL FOR BEGINNERS

head



tail



cat

Opisy na kolejnych slajdach będą pochodzić z wydania 5.10 podręczników man.

Część I

# Informacje wstępne

# Przygotowanie skryptów

- Na zajęciach będziemy pisać tak zwane skrypty powłoki Bash.
- Będą to po prostu pliki, zawierające zestawy komend...
- Te same, których moglibyśmy użyć ręcznie w trybie interaktywnym.
- Do pisania skryptów można użyć dowolnego edytora plików tekstowych.
  - ▶ Na przykład **VIM** lub **Visual Studio Code**.
- Do uruchomienia skryptów potrzebny jest emulator terminala (konsola).
- Przykładowy skrypt:

```
#!/usr/bin/env bash

katalog='nowy'

if [ ! -d "${katalog}" ]; then
    mkdir "${katalog}"
fi

echo 'Skończyłem!'
```

# Uruchamianie i argumenty skryptów

- Przygotowane skrypty możemy uruchomić analogicznie jak inne programy.
- W tym – nasze skrypty mogą przyjmować różne argumenty.
- Przykładowe wywołania:

- ▶ `bash skrypt.sh argument1 argument2 'argument 3'`,
- ▶ `./skrypt.sh argument1 argument2 'argument 3'`.

- Przykładowa obsługa argumentów:

```
#!/usr/bin/env bash
```

```
echo $# # liczba argumentów skryptu (-argc)
```

```
echo $@ # wszystkie przekazane argumenty (-argv)
```

```
echo $0 # nazwa bieżącego skryptu (jak został uruchomiony)
```

```
echo $1 # pierwszy argument skryptu: argument1
```

```
echo $2 # drugi argument skryptu: argument2
```

```
echo $3 # trzeci argument skryptu: argument 3 <--- ze spacją! (bo apostrofy...)
```

## Część II

# Podstawowe komendy

# Najważniejsze na początek – program man

- Narzędzie do szybkiego wyszukiwania opisu/pomocy/składni.
- Przykłady użycia:
  - ▶ `man bash`
  - ▶ `man head`
  - ▶ `man hier`

## NAME

`man` - an interface to the system reference manuals

## SYNOPSIS

```
man [man options] [[section] page ...] ...  
man -k [apropos options] regexp ...  
man -K [man options] [section] term ...  
man -f [whatis options] page ...  
man -l [man options] file ...  
man -w|-W [man options] page ...
```

## DESCRIPTION

`man` is the system's manual pager. Each page argument given to `man` is normally the name of a program, utility or function. The manual page associated with each of these arguments is then found and displayed. A section, if provided, will direct `man` to look only in that section of the manual. The default action is to search in all of the available sections following a pre-defined order (see `DEFAULTS`), and to show only the first page found, even if page exists in several sections.

# Program cat

- Narzędzie do wyświetlania zawartości jednego lub kilku plików po kolei.
- Przykłady użycia:
  - ▶ `cat /ścieżka/do/jakiegoś/pliku`
  - ▶ `cat plik1 plik2 plik3`

## NAME

`cat` - concatenate files and print on the standard output

## SYNOPSIS

`cat [OPTION]... [FILE]...`

## DESCRIPTION

Concatenate `FILE(s)` to standard output.

With no `FILE`, or when `FILE` is `-`, read standard input.

`-A, --show-all`  
equivalent to `-vET`

`-b, --number-nonblank`  
number nonempty output lines, overrides `-n`

`-e` equivalent to `-vE`



# Program head

- Narzędzie służące do wyświetlenia linii z początku podanych plików.
- Przykłady użycia:
  - ▶ `head -n 10 plik` *# tylko 10 pierwszych linii*
  - ▶ `head -n -5 plik` *# wszystko prócz 5 ostatnich linii*

## NAME

`head` - output the first part of files

## SYNOPSIS

`head [OPTION]... [FILE]...`

## DESCRIPTION

Print the first 10 lines of each FILE to standard output. With more than one FILE, precede each with a header giving the file name.

With no FILE, or when FILE is `-`, read standard input.

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

`-c, --bytes=[-]NUM`

print the first NUM bytes of each file; with the leading `'-'`, print all but the last NUM bytes of each file

`-n, --lines=[-]NUM`

print the first NUM lines instead of the first 10; with the leading `'-'`, print all

# Program tail

– Narzędzie służące do wyświetlenia linii z końca podanych plików.

– Przykłady użycia:

▶ `tail -n 10 plik` *# tylko 10 ostatnich linii*

▶ `tail -n +5 plik` *# wszystko prócz 5 pierwszych linii*

## NAME

`tail` - output the last part of files

## SYNOPSIS

`tail [OPTION]... [FILE]...`

## DESCRIPTION

Print the last 10 lines of each FILE to standard output. With more than one FILE, precede each with a header giving the file name.

With no FILE, or when FILE is `-`, read standard input.

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

`-c, --bytes=[+]NUM`

output the last NUM bytes; or use `-c +NUM` to output starting with byte NUM of each file

`-f, --follow[={name|descriptor}]`

output appended data as the file grows;

# Program ls

- Narzędzie do listowania zawartości wskazanego katalogu.

- Przykłady użycia:

- ▶ `ls` *# wypisze pliki z bieżącego katalogu roboczego*

- ▶ `ls ścieżka/` *# wypisze zawartość katalogu ścieżka/*

## 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` is specified.

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 `..`

`--author`

with `-l`, print the author of each file

# Program cd

- Narzędzie do zmienia bieżącego katalogu roboczego (\$CWD).
- Przykłady użycia:

▶ `cd # przenosi do katalogu domowego ($HOME)`

▶ `cd ścieżka/ # przenosi do katalogu ścieżka/`

## NAME

`cd -- change the working directory`

## SYNOPSIS

`cd [-L|-P] [directory]`

`cd -`

## DESCRIPTION

The `cd` utility shall change the working directory of the current shell execution environment (see Section 2.12, Shell Execution Environment) by executing the following steps in sequence. (In the following steps, the symbol `curpath` represents an intermediate value used to simplify the description of the algorithm used by `cd`. There is no requirement that `curpath` be made visible to the application.)

1. If no directory operand is given and the `HOME` environment variable is empty or undefined, the default behavior is implementation-defined and no further steps shall be taken.
2. If no directory operand is given and the `HOME` environment variable is set to a non-

# Program pwd

- Narzędzie do wyświetlenia bieżącego katalogu roboczego.
- Przykłady użycia:

► **pwd**

## NAME

`pwd` - print name of current/working directory

## SYNOPSIS

`pwd [OPTION]...`

## DESCRIPTION

Print the full filename of the current working directory.

`-L, --logical`

use PWD from environment, even if it contains symlinks

`-P, --physical`

avoid all symlinks

`--help` display this help and exit

`--version`

output version information and exit

If no option is specified, `-P` is assumed.

# Program echo

- Narzędzie do wyświetlania przekazanych argumentów (tekstu).
- Przykłady użycia:

▶ `echo "w4rto $RANDOM razy"`

▶ `echo -n 'tutaj bez dodatkowego \n na końcu'`

## NAME

`echo` - display a line of text

## SYNOPSIS

`echo [SHORT-OPTION]... [STRING]...`

`echo LONG-OPTION`

## DESCRIPTION

Echo the STRING(s) to standard output.

`-n` do not output the trailing newline

`-e` enable interpretation of backslash escapes

`-E` disable interpretation of backslash escapes (default)

`--help` display this help and exit

`--version`

output version information and exit

# Program touch

- Narzędzie do tworzenia nowych plików i zmiany daty ich otwarcia.
- Przykłady użycia:

► `touch plik`

## NAME

`touch` - change file timestamps

## SYNOPSIS

`touch [OPTION]... FILE...`

## DESCRIPTION

Update the access and modification times of each FILE to the current time.

A FILE argument that does not exist is created empty, unless `-c` or `-h` is supplied.

A FILE argument string of `-` is handled specially and causes touch to change the times of the file associated with standard output.

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

`-a` change only the access time

`-c, --no-create`  
do not create any files

`-d, --date=STRING`

# Program mv

- Narzędzie do zmieniania nazw i przenoszenia plików.

- Przykłady użycia:

- ▶ `mv stary nowy # zmiana nazwy pliku`

- ▶ `mv plik1 plik2 plik3 katalog/ # przeniesienie`

## NAME

`mv` - move (rename) files

## SYNOPSIS

`mv [OPTION]... [-T] SOURCE DEST`

`mv [OPTION]... SOURCE... DIRECTORY`

`mv [OPTION]... -t DIRECTORY SOURCE...`

## DESCRIPTION

Rename SOURCE to DEST, or move SOURCE(s) to DIRECTORY.

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

`--backup[=CONTROL]`

make a backup of each existing destination file

`-b` like `--backup` but does not accept an argument

`-f, --force`

do not prompt before overwriting



# Program cp

- Narzędzie do kopiowania plików.
- Przykłady użycia:
  - ▶ `cp źródło kopia # pojedynczy plik`
  - ▶ `cp plik1 plik2 plik3 katalog-docelowy/ # kilka na raz`

## NAME

`cp` - copy files and directories

## SYNOPSIS

```
cp [OPTION]... [-T] SOURCE DEST
cp [OPTION]... SOURCE... DIRECTORY
cp [OPTION]... -t DIRECTORY SOURCE...
```

## DESCRIPTION

Copy SOURCE to DEST, or multiple SOURCE(s) to DIRECTORY.

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

`-a, --archive`

same as `-dR --preserve=all`

`--attributes-only`

don't copy the file data, just the attributes

`--backup[=CONTROL]`

# Program rm

- Narzędzie do usuwania plików i katalogów. **Zachować ostrożność!**
- Przykłady użycia:
  - ▶ `rm coś` *# pojedynczy plik*
  - ▶ `rm -r ścieżka/` *# usunięcie katalogu, ostrożnie!!!*

## NAME

`rm` - remove files or directories

## SYNOPSIS

`rm [OPTION]... [FILE]...`

## DESCRIPTION

This manual page documents the GNU version of `rm`. `rm` removes each specified file. By default, it does not remove directories.

If the `-I` or `--interactive=once` option is given, and there are more than three files or the `-r`, `-R`, or `--recursive` are given, then `rm` prompts the user for whether to proceed with the entire operation. If the response is not affirmative, the entire command is aborted.

Otherwise, if a file is unwritable, standard input is a terminal, and the `-f` or `--force` option is not given, or the `-i` or `--interactive=always` option is given, `rm` prompts the user for whether to remove the file. If the response is not affirmative, the file is skipped.

# Program mkdir

- Narzędzie do tworzenia katalogów.
- Przykłady użycia:
  - ▶ `mkdir ścieżka/` *# pojedynczy katalog*
  - ▶ `mkdir -p katalog/i/jakieś/podkatalogi` *# od razu drzewo*

## NAME

`mkdir - make directories`

## SYNOPSIS

`mkdir [OPTION]... DIRECTORY...`

## DESCRIPTION

Create the DIRECTORY(ies), if they do not already exist.

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

`-m, --mode=MODE`

set file mode (as in `chmod`), not `a=rwx - umask`

`-p, --parents`

no error if existing, make parent directories as needed

`-v, --verbose`

print a message for each created directory

# Program rmdir

- Narzędzie do usuwania pustych (tylko i wyłącznie) katalogów.
- Przykłady użycia:

▶ `rmdir ścieżka/` *# pojedynczy pusty katalog*

▶ `rmdir -p katalog/i/podkatalogi` *# kilka pustych na raz*

## NAME

`rmdir` - remove empty directories

## SYNOPSIS

`rmdir` [OPTION]... DIRECTORY...

## DESCRIPTION

Remove the DIRECTORY(ies), if they are empty.

`--ignore-fail-on-non-empty`

ignore each failure that is solely because a directory

is non-empty

`-p, --parents`

remove DIRECTORY and its ancestors; e.g., `'rmdir -p a/b/c'` is similar to `'rmdir a/b/c a/b a'`

`-v, --verbose`

## Część III

# Przydatne konstrukcje

# Testowanie plików i porównywanie wartości

- Klasycznie używało się do tego programu `test`.
  - ▶ W powłoce Bash używany jest wbudowany mechanizm zamiast tego.
  - ▶ Jest on jednak kompatybilny z oryginalnym programem `test`.
  - ▶ Czyli strona podręcznika do programu `test` nadal jest w cenie!
- Przykłady użycia programu `test` – np. w ramach instrukcji warunkowej:
  - ▶ `test` wyrażenie    `- if test -e plik; then ... fi`
  - ▶ `[ wyrażenie ]`    `- if [ -e plik ]; then ... fi`
- Przykładowe najważniejsze wyrażenia:
  - ▶ `-e ścieżka` – obiekt ścieżka istnieje,
  - ▶ `-d ścieżka` – obiekt ścieżka istnieje i jest katalogiem,
  - ▶ `-f ścieżka` – obiekt ścieżka istnieje i jest zwykłym plikiem,
  - ▶ `-r ścieżka` – obiekt ścieżka istnieje i możemy go przeczytać,
  - ▶ `-w ścieżka` – obiekt ścieżka istnieje i możemy go nadpisać,
  - ▶ `-x ścieżka` – obiekt ścieżka istnieje i możemy go uruchomić,
  - ▶ `napis1 = napis2` – porównanie wartości łańcuchów znaków,
  - ▶ `! wyrażenie` – negacja wyniku wyrażenia (np. `! -e plik`).

# Przechwytywanie i zapis wyników programów

- Zdarza się, że musimy wykonać coś, na podstawie zawartości pliku.
  - ▶ Zawartość pliku można przeczytać, np. komendą `cat plik`.
  - ▶ Istnieje mechanizm, pozwalający zapisać wynik komend do zmiennej.
  - ▶ Przykład zapisu: `zmienna=$( komenda )`.<sup>1</sup>
- Wynik działania komendy można zawsze zapisać do pliku.
  - ▶ Nawet, jeśli używany program domyślnie wyświetla wynik na ekranie.
  - ▶ Wystarczy na końcu komendy dodać: `> ścieżka-do-pliku`.
  - ▶ Jest to wtedy tzw. przekierowanie wyjścia.
  - ▶ Przykład: `echo -e "$HOME\n$PATH" > plik`.
  - ▶ **Uwaga!** Domyślnie nadpisze to bieżącą zawartość istniejącego pliku!
  - ▶ Operator `>>` pozwala dopisać nową treść dopisać na końcu pliku.
  - ▶ Przykład: `echo "$USER" >> plik`.

---

<sup>1</sup> Istnieje jeszcze wariant z wykorzystaniem tzw. odwróconych apostrofów (``` - ang. *backtick*, na klawiaturze nad klawiszem [TAB]), ale raczej nie zaleca się jego stosowania, ze względu na różne jego ograniczenia.

# Obliczenia w powłoce

- W powłoce możliwe jest wykonywanie podstawowych obliczeń.
- Dawniej stosowano w tym celu po prostu program `expr`.
- W powłoce Bash dostępny jest wygodniejszy mechanizm: komenda `let`.
  - ▶ Ma on też alternatywny zapis: `$( ( wyrażenie ) )`.
  - ▶ W wyrażeniach nie stosujemy znaku `$` przed nazwami zmiennych.
  - ▶ Nie mają też znaczenia białe znaki pomiędzy operatorami.
- Warto pamiętać o specjalnej zmiennej `$RANDOM`.
- Przykłady użycia:
  - ▶ `let "wynik = 89 * 13"`
  - ▶ `wynik=$( ( RANDOM % 6 ) )`
  - ▶ Sprawdzenie zawartości: `echo "${wynik}"`.
- Standardem są wyłącznie obliczenia dla liczb całkowitych.
  - ▶ Niektóre powłoki wspierają też obliczenia dla liczb wymiernych.
  - ▶ Dla większej precyzji można posłużyć się na przykład programem `bc`.



# Jak się uczyć bez obaw?

- Część komend, niepoprawnie użyta, może skutkować utratą danych.
  - ▶ W szczególności chodzi tutaj o program `rm`.
  - ▶ Także programy `mv` i `cp`, jeśli użyć ich z rozmysłem.
  - ▶ Zwykły zapis do plików (`>` i `>>`) to również ryzyko utraty danych.
- Naukę lepiej przeprowadzić w środowisku bez ważnych dokumentów.
  - ▶ Nie, osobny katalog na dysku może nie wystarczyć!
  - ▶ Maszyna wirtualna sprawdzi się świetnie.
  - ▶ Można też użyć technologii kontenerowych (np. Docker, podman).
- Poza tym **zawsze** sprawdzaj w którym terminalu pracujesz.
- Zawsze też pilnuj zawartości schowka i nie wklejaj w terminal bez uwagi!

# Przykłady użycia środowiska Docker

- ▶ Utworzenie nowego kontenera:

```
docker run -it ubuntu:focal /bin/bash
```

- ▶ Podłączenie się do działającego kontenera:

```
docker exec -it ID_KONTENERA /bin/bash
```

- ▶ Wylistowanie wszystkich dostępnych w systemie kontenerów:

```
docker ps -a
```

- ▶ Uruchomienie/wyłączenie istniejącego w systemie kontenera:

```
docker start ID_KONTENERA
```

```
docker stop ID_KONTENERA
```

- ▶ Usunięcie istniejącego w systemie kontenera:

```
docker rm ID_KONTENERA
```

**Uwaga!** Wiele standardowych obrazów kontenerów posiada tylko jednego roboczego użytkownika root. Domyślnie jest to właściciel maszyny i z racji tego ma on zawsze rozszerzone uprawnienia. Z perspektywy naszego kursu **zadania należy wykonywać jako zwykły użytkownik** – poniższa komenda pozwala stworzyć takiego o nazwie lab:

- ▶ `useradd --user-group --create-home --shell /bin/bash lab`

Przełączenie się na konto użytkownika lab:

- ▶ `su -l lab` *# bez opcji -l trzeba ręcznie wejść do katalogu domowego*

Część dla dociekliwych

# Zadanie dodatkowe

# Zadanie domowe – dla chętnych

Bash wspiera wiele użytecznych mechanizmów związanych ze zmiennymi (czasem ogólnie nazywanych parametrami). Obejmują one takie operacje, jak podstawienie domyślnej wartości, wycięcie fragmentu zawartości zmiennej, podmianę wartości, itp.

Proszę odnaleźć kilka<sup>1</sup> przykładowych mechanizmów (w szczególności takich, które wydadzą się najbardziej przydatne), omówić je i podać przykłady ich użycia.

Hasło pomocnicze do wyszukania: **Shell Parameter Expansion**.

Chodzi tutaj o wszelkie zapisy typu `${zmienna:cośtam}`.

---

<sup>1</sup> pięć wystarczy ;-)