Zaawansowane metody programowania

Dr Zbigniew Kozioł - wykład
Mgr Mariusz Woźny - laboratorium

Wykład III

Elementy programowania

- operacje wejścia/wyjścia
- STD INPUT, STD OUTPUT, STD ERROR
- grep, przekierowywanie danych, perl i bash, gnuplot
- | (pipe)

Filozofia programowania (Unix)

The Unix philosophy, originated by Ken Thompson, is a set of cultural norms and philosophical approaches to developing small yet capable software based on the experience of leading developers of the Unix operating system. Early Unix developers were important in bringing the concepts of modularity and reusability into software engineering practice, spawning a "software tools" movement. Over time, the leading developers of Unix (and programs that ran on it) established a set of cultural norms for developing software, norms which became as important and influential as the technology of Unix itself; this has been termed the "Unix philosophy."

The Unix philosophy emphasizes building short, simple, clear, modular, and extensible code that can be easily maintained and repurposed by developers other than its creators. The Unix philosophy favors composability as opposed to monolithic design.

Filozofia programowania (Unix); Mike Gancarz, 1994

1. Small is beautiful.
2. Make each program do one thing well.
3. Build a prototype as soon as possible.
4. Choose portability over efficiency.
5. Store data in flat text files.
6. Use software leverage to your advantage.
7. Use shell scripts to increase portability.
8. Avoid captive user interfaces.
9. Make every program a filter.

STDINPUT, STDOUTPUT, STDERROR

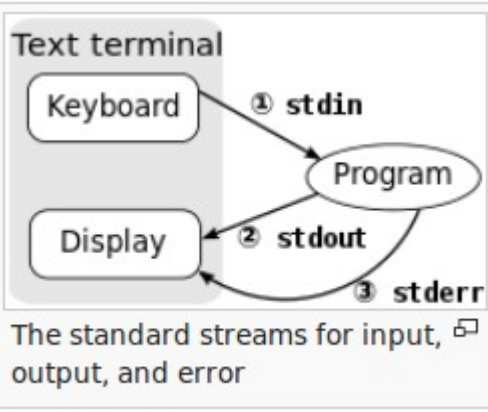
Standardowe strumienie wejścia, wyjścia i błędu

STDINPUT, STDOUTPUT, STDERROR

In computer programming, standard streams are preconnected input and output communication channels between a computer program and its environment when it begins execution. The three I/O connections are called standard input (stdin), standard output (stdout) and standard error (stderr). Originally I/O happened via a physically connected system console (input via keyboard, output via monitor), but standard streams abstract this.

When a command is executed via a shell, the streams are typically connected to the text terminal in which the shell is running, but can be changed with redirection, particularly via a pipeline.

Standardowe strumienie – standardowe kanały komunikacji między komputerem a otoczeniem (zwykle terminalem). Występują w Uniksie i systemach uniksopodobnych, w środowisku uruchomieniowym C, C++ i ich pochodnych. Trzy podstawowe połączenia I/O noszą nazwy: standard input (stdin, standardowy strumień wejścia), standard output (stdout, standardowy strumień wyjścia) i standard error (stderr, standardowy strumień błędów).



The standard streams for input, output, and error

Standardowy strumień wejścia

Standardowy strumień wejścia to dane (zwykle tekst) przekazywane do programu. Pobiera on dane poprzez użycie funkcji read. Nie wszystkie programy wymagają danych wejściowych.

```
bash < skrypt.sh        # spowoduje wykonanie instrukcji zawartych w
                        # pliku skrypt.sh przez interpretator bash
cat < plik.txt          # taki sam efekt jak cat plik.txt
```

Albo taki przykład. Plik tekstowy my_wake.sh o następującej zawartości:

```
/usr/bin/play "/usr/share/webkitgtk-1.0/resources/audio/Composite.wav"
/usr/bin/play "/usr/share/sounds/alsa/Front_Center.wav"
/usr/bin/play "/usr/share/sounds/speech-dispatcher/test.wav"
/usr/bin/play "/usr/share/sounds/alsa/Side_Right.wav"
/usr/bin/play "/home/zbych/.config/libreoffice/4/user/gallery/bomb.wav"
```

Plik uruchamiam komendą:

```
at -f /home/zbych/my_wake.sh 07:00 tomorrow
```

Pytanie: gdzie tu są standardowe strumienie wejścia?

Standardowy strumień wyjścia

Standardowy strumień wyjścia to strumień, do którego program zapisuje dane wynikowe. Robi to za pomocą funkcji write. Niektóre programy nie zwracają danych wynikowych.

```
echo "ABC" > plik.txt      # spowoduje wyczyszczenie pliku plik.txt
                           # (lub utworzenie jeśli nie istnieje)
                           # i dopisanie znaków  ABC

echo "CBA" >> plik.txt     # spowoduje dopisanie znaków  CBA na koniec
                           # pliku plik.txt
                           # lub jego utworzenie i dopisanie tych znaków

cat plik1.txt > plik2.txt  # efekt identyczny jak w przypadku polecenia
                           # cp plik1.txt plik2.txt
```

Standardowy strumień błędów

Standardowy strumień błędów jest zwykle wykorzystywany do wyświetlania komunikatów o błędach i informacjach przydatnych do debugowania.

Jest on niezależny od strumienia wyjścia.

Zwykle celem strumienia jest, podobnie jak przy stdout terminal z którego uruchomiono program aby umożliwić zobaczenie błędu nawet wtedy, gdy strumień wyjścia jest przekierowany.

Jeżeli używamy potoku aby użyć danych wynikowych jakiegoś programu jako danych wejściowych dla innego to błędy i tak zostaną wypisane na terminalu.

```
$ rm -rf /tmp  # spowoduje usunięcie wszystkich plików z katalogu /tmp,
               # do których dany użytkownik ma prawa pisania ("w") oraz
               # wyświetlenie komunikatów o tym, że nie można usunąć
               # tych plików, do których użytkownik nie posiada praw

$ rm -rf /tmp 2> /dev/null  # spowoduje usunięcie wszystkich plików
               # z katalogu /tmp, do których dany użytkownik ma prawa
               # pisania ("w"), a komunikaty o błędach zostaną
               # przekierowane do /dev/null, czyli Śmietnika
```

Potoki

| Pipe, pipeline (ang); Potok (PL)

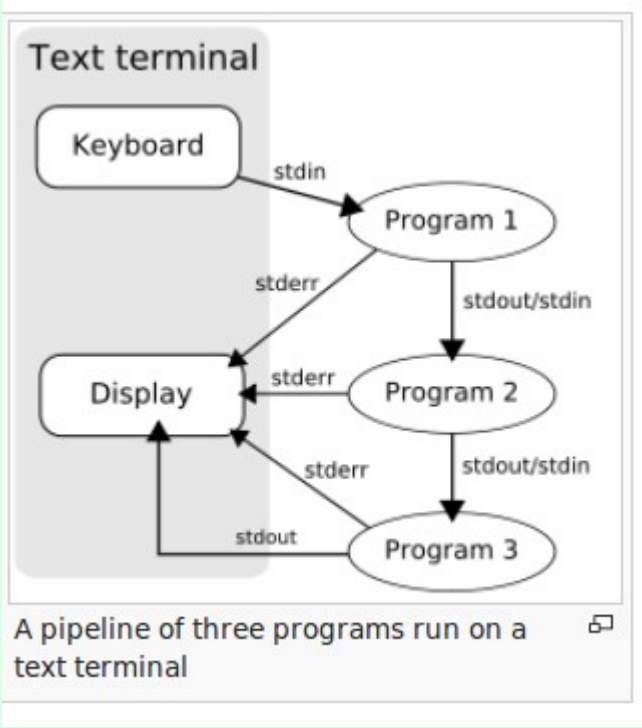
Potok – jeden z mechanizmów komunikacji międzyprocesowej umożliwiający wymianę danych pomiędzy dwoma procesami. Odbywa się to najczęściej poprzez połączenie standardowego wyjścia jednego procesu ze standardowym wejściem drugiego.

In Unix-like computer operating systems (and, to some extent, Microsoft Windows), a pipeline is a set of processes chained by their standard streams, so that the output of each process (stdout) feeds directly as input (stdin) to the next one. Filter programs are often used in this configuration.

The concept of pipelines was invented by Douglas McIlroy at Unix's ancestral home of Bell Labs, prior to the actual invention of the operating system, and implemented in Unix at his insistence, shaping its toolbox philosophy. It is named by analogy to a physical pipeline. The standard shell syntax for pipelines is to list multiple programs to invoke in one command, separated by vertical bars:

```
program1 | program2 | program3
```

| Pipe, pipeline (ang); Potok (PL), c.d.



A pipeline of three programs run on a text terminal

Przykład wykorzystania potoków w systemie UNIX:

```
$ ps -a | sort | uniq | grep -v sh
```

Powyższa konstrukcja zwróci listę uruchomionych procesów (ps -a), posortowaną alfabetycznie (sort), niezawierającą powtórzeń (uniq), oraz bez linii zawierających wzorzec sh (grep -v sh).

| Pipe, pipeline, więcej...

In most Unix-like systems, all processes of a pipeline are started at the same time, with their streams appropriately connected, and managed by the **scheduler** together with all other processes running on the machine.

An important aspect of this, setting Unix pipes apart from other pipe implementations, is the concept of **buffering**: for example a sending program may produce 5000 bytes per second, and a receiving program may only be able to accept 100 bytes per second, but no data is lost. Instead, the output of the sending program is held in a **queue**.

When the receiving program is ready to read data, the operating system sends its data from the queue, then removes that data from the queue. If the queue buffer fills up, the sending program is suspended (blocked) until the receiving program has had a chance to read some data and make room in the buffer.

Grep

```
zbych@orion:~$ grep --help
Usage: grep [OPTION]... PATTERN [FILE]...
Search for PATTERN in each FILE or standard input.
PATTERN is, by default, a basic regular expression (BRE).
Example: grep -i 'hello world' menu.h main.c

Regexp selection and interpretation:
  -E, --extended-regexp     PATTERN is an extended regular expression (ERE)
  -F, --fixed-strings       PATTERN is a set of newline-separated fixed strings
  -G, --basic-regexp        PATTERN is a basic regular expression (BRE)
  -P, --perl-regexp         PATTERN is a Perl regular expression
  -e, --regexp=PATTERN      use PATTERN for matching
  -f, --file=FILE           obtain PATTERN from FILE
  -i, --ignore-case         ignore case distinctions
  -w, --word-regexp         force PATTERN to match only whole words
  -x, --line-regexp         force PATTERN to match only whole lines
  -z, --null-data           a data line ends in 0 byte, not newline

Miscellaneous:
  -s, --no-messages         suppress error messages
  -v, --invert-match        select non-matching lines
  -V, --version             print version information and exit
      --help                display this help and exit
      --mmap                deprecated no-op; evokes a warning
```

Grep, więcej...

```
Output control:
  -m, --max-count=NUM       stop after NUM matches
  -b, --byte-offset         print the byte offset with output lines
  -n, --line-number         print line number with output lines
      --line-buffered       flush output on every line
  -H, --with-filename       print the file name for each match
  -h, --no-filename         suppress the file name prefix on output
      --label=LABEL         use LABEL as the standard input file name prefix
  -o, --only-matching       show only the part of a line matching PATTERN
  -q, --quiet, --silent     suppress all normal output
      --binary-files=TYPE   assume that binary files are TYPE;
                            TYPE is 'binary', 'text', or 'without-match'
  -a, --text                equivalent to --binary-files=text
  -I                        equivalent to --binary-files=without-match
  -d, --directories=ACTION  how to handle directories;
                            ACTION is 'read', 'recurse', or 'skip'
  -D, --devices=ACTION      how to handle devices, FIFOs and sockets;
                            ACTION is 'read' or 'skip'
  -r, --recursive           like --directories=recurse
  -R, --dereference-recursive  likewise, but follow all symlinks
      --include=FILE_PATTERN  search only files that match FILE_PATTERN
      --exclude=FILE_PATTERN  skip files and directories matching FILE_PATTERN
      --exclude-from=FILE   skip files matching any file pattern from FILE
      --exclude-dir=PATTERN  directories that match PATTERN will be skipped.
  -L, --files-without-match  print only names of FILEs containing no match
  -l, --files-with-matches  print only names of FILEs containing matches
  -c, --count               print only a count of matching lines per FILE
  -T, --initial-tab         make tabs line up (if needed)
  -Z, --null                print 0 byte after FILE name
```

Grep, jeszcze więcej...

```
Context control:
  -B, --before-context=NUM  print NUM lines of leading context
  -A, --after-context=NUM   print NUM lines of trailing context
  -C, --context=NUM         print NUM lines of output context
  -NUM                      same as --context=NUM
      --color[=WHEN],
      --colour[=WHEN]       use markers to highlight the matching strings;
                            WHEN is 'always', 'never', or 'auto'
  -U, --binary              do not strip CR characters at EOL (MSDOS/Windows)
  -u, --unix-byte-offsets   report offsets as if CRs were not there
                            (MSDOS/Windows)

'egrep' means 'grep -E'.  'fgrep' means 'grep -F'.
Direct invocation as either 'egrep' or 'fgrep' is deprecated.
When FILE is -, read standard input.  With no FILE, read . if a command-line
-r is given, - otherwise.  If fewer than two FILEs are given, assume -h.
Exit status is 0 if any line is selected, 1 otherwise;
if any error occurs and -q is not given, the exit status is 2.

Report bugs to: bug-grep@gnu.org
GNU Grep home page: <http://www.gnu.org/software/grep/>
General help using GNU software: <http://www.gnu.org/gethelp/>
```

Grep, przykłady

Wyrażenia regularne
(Regular expressions)

^

$

- Stosowane bywają programistycznie
- Ale też w komendach takich jak grep