Kurs administrowania systemem Linux

Zajęcia nr 1: Powłoka systemowa

Instytut Informatyki Uniwersytetu Wrocławskiego

27 lutego 2025

Filozofia Uniksa

Idea klocków Lego

1. Duży zbiór małych, wyspecjalizowanych programów, z których każdy dobrze i efektywnie wykonuje jedno określone zadanie.
2. Elastyczne narzędzia pozwalające na zestawianie programów w dowolnie duże konstelacje.

Rozwiązanie w Uniksie

1. Np. `ls`, `stat`, `cp`, `mv`, `rm`, `mkdir`, `rmdir`, `chmod`, `chown`, `du`, `df` i dziesiątki innych zamiast jednego wielkiego programu *File Explorer*.
2. *Strumienie* (`stdin`, `stdout` i `stderr`) oraz *potoki* plus mechanizm przekazywania parametrów do programów.

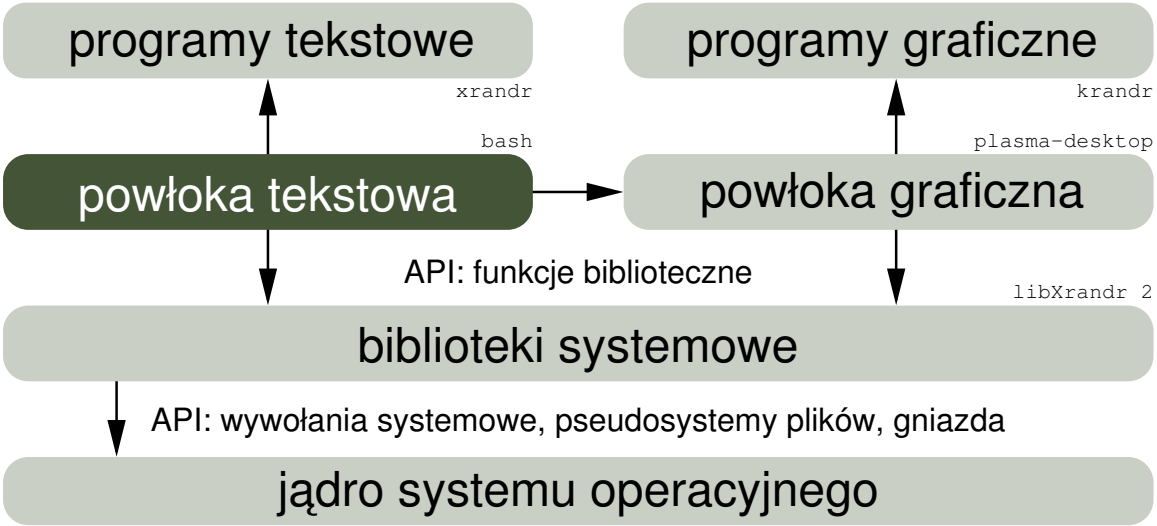
Centralnym narzędziem służącym do uruchamiania i zestawiania programów ze sobą jest *powłoka systemowa*.

Filozofia Uniksa nie jest uniwersalna

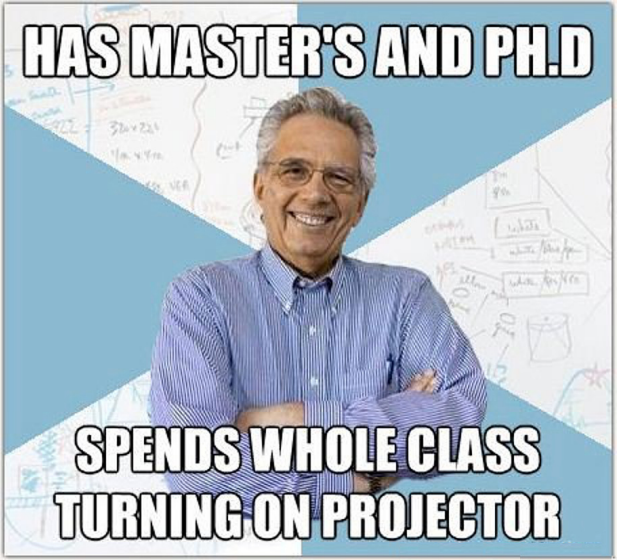
Czasem dekompozycja zadania na małe, niezależne fragmenty jest trudna, niewygodna lub niemożliwa.

- SysV Init + rsyslog + cron + ... vs. SystemD
- ext4 + LVM2 + dm-raid + ... vs. ZFS
- mikrojądro vs. jądro monolityczne
- X + Openbox + Urxvt + ... vs. Gnome

Powłoka tekstowa w systemach uniksowych



Konfiguracja monitora w praktyce



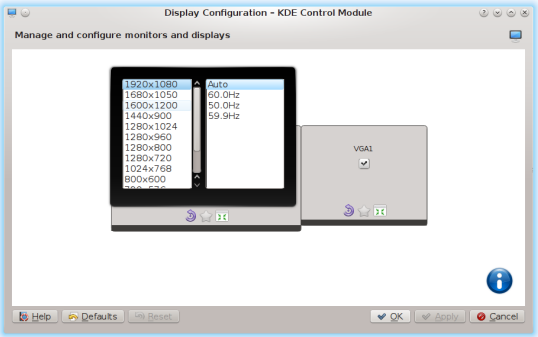
Programy tekstowe i graficzne. Przykład: xrandr i krandr

Program tekstowy

```
... 1-$ scrot  2*$ man  8@$ msg  ...          Mon Nov 12 18:38 BAT100% SCR30%
XRANDR(1)                General Commands Manual                XRANDR(1)

NAME
       xrandr - primitive command line interface to RandR extension

SYNOPSIS
       xrandr  [--help]  [--display display] [-q] [-v] [--verbose] [--dryrun]
       [--screen snum] [--q1]  [--q12]  [--current]  [--noprimary]  [--panning
       widthxheight[+x+y[/track_widthxtrack_height+track_x+track_y[/bor-
       der_left/border_top/border_right/border_bottom]]]]    [--scale     xxy]
       [--scale-from wxh] [--transform a,b,c,d,e,f,g,h,i] [--primary] [--prop]
       [--fb widthxheight] [--fbmm widthxheight] [--dpi dpi]  [--newmode  name
       mode]  [--rmmode  name] [--addmode output name] [--delmode output name]
       [--output output] [--auto]  [--mode  mode]  [--preferred]  [--pos  xxy]
       [--rate  rate] [--reflect reflection] [--rotate orientation] [--left-of
       output] [--right-of output] [--above output] [--below output]  [--same-
       as  output]  [--set  property  value]  [--off]  [--crtc  crtc] [--gamma
       red:green:blue] [--brightness brightness] [--o  orientation]  [-s  size]
       [-r   rate]   [-x]  [-y]  [--listproviders]  [--setprovideroutputsource
       provider source] [--setprovideroffloadsink provider sink]
Manual page xrandr(1) line 1 (press h for help or q to quit)
```

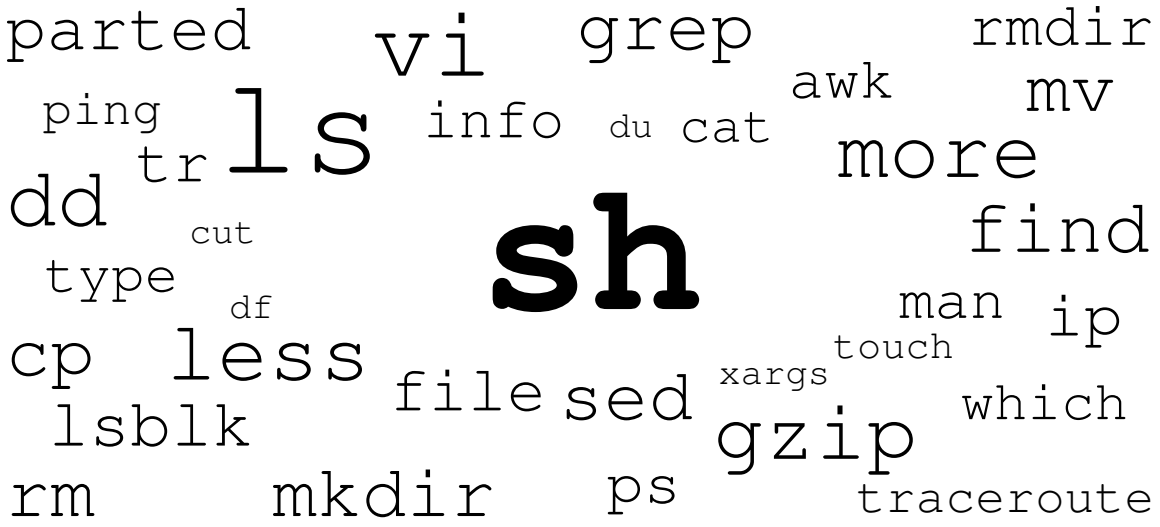
- Udostępnia wszelkie zaimplementowane usługi.
- Duża niezawodność i powtarzalność działania.
- Duże możliwości diagnostyki w razie problemów.
- Łatwa automatyzacja (skryptowanie).
- Wymaga przeczytania podręcznika obsługi.

Program graficzny

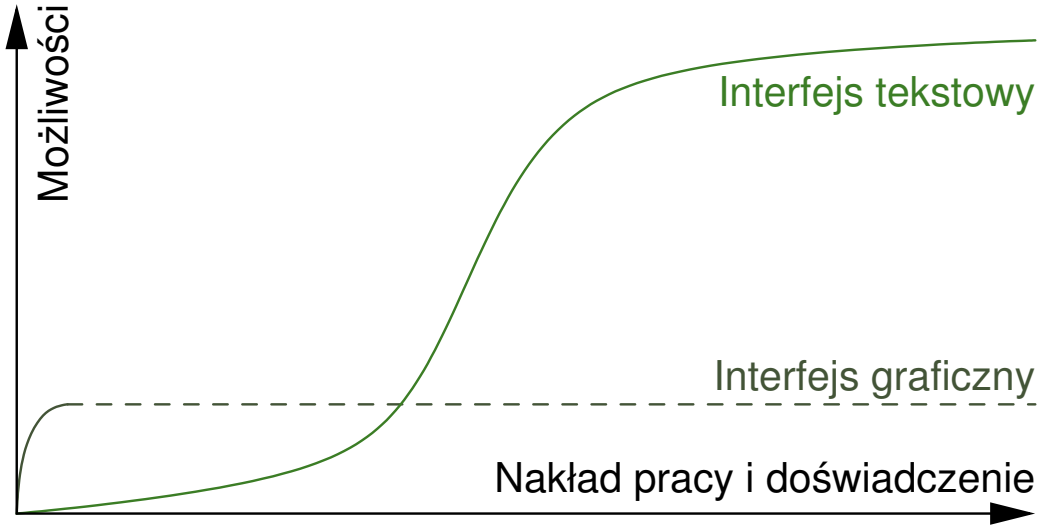


- Udostępnia niewielki podzbiór usług.
- Zawodność i niepowtarzalność działania.
- Niewielkie możliwości diagnostyki.
- Brak możliwości automatyzacji.
- Intuicyjny interfejs, bez potrzeby nauki.

Filozofia Uniksa



Nakład pracy i doświadczenie



Powłoki w Linuksie

sh i kompatybilne

- sh (Bourne 1979)
- ash (Almquist)
- **bash** (Bourne Again)
- dash (Debian Almquist)
- ksh (Korn)
- mksh (MirBSD Korn)
- zsh (Zhong Shao)
- busybox

csh i kompatybilne

- csh (Berkeley C)
- tcsh (Tenex C)

Inne, egzotyczne

- yash (Yet Another)
- scsh (Scheme)

Najpopularniejszą powłoką w Linuksie jest bash.

Dokumentacja GNU Bash

Dwa podobne, ale różne dokumenty:

- `bash(1)` (*Linux man page*, źródło: `groff`),
- *Bash Reference Manual*, Chet Ramey (CWRU), Brian Fox (FSF) (GNU Texinfo, źródło: `texi`).

Slang: „man bash" — długi i niezrozumiały dokument.
Nic bardziej mylnego!

Warto też czytać podręczniki, np.:

- *bash Cookbook*, Carl Albing, JP Vossen, Cameron Newham, O'Reilly 2007. Pol. tłum.: *bash. Receptury*, Helion 2008.
- *Learning the bash Shell*, Cameron Newham, Bill Rosenblatt, O'Reilly, 3rd ed. 2005.
- *Bash Pocket Reference*, Arnold Robbins, O'Reilly, 2nd ed. 2016.

. . . i wiele innych.

Dokumentacja GNU Bash w PDF-ie

Dokumentację można oglądać na ekranie:

- `man bash`
- `info bash`

ale wygodniej jest skonwertować do PDF-a, wydrukować i czytać do poduszki:

```
MANROFFOPT="-f H -T ps -rS12 -e -mandoc"
{ ps2ps <(man -t bash) >(psbook | pstops -p a4 \
  "2:0L@.76(22.2cm,-1.2cm)+1L@.76(22.2cm,14.35cm)" | \
  ps2pdf - > bash.pdf)
} | cat
```

Bash Reference Manual jest zwykle dostępny w PDF-ie w rozmiarze letter (pakiet `bash-doc`):

```
/usr/share/doc/bash/bashref.pdf
```

ale warto samodzielnie wygenerować A4:

```
texi2pdf --command=@afourpaper bashref.texi
```

man bash w PDF-ie

BASH(1) General Commands Manual BASH(1)

NAME

bash − GNU Bourne-Again SHell

SYNOPSIS

bash [options] [command_string | file]

COPYRIGHT

DESCRIPTION

Bash is an **sh**-compatible command language interpreter that executes commands read from the standard input or from a file. **Bash** also incorporates useful features from the *Korn* and *C* shells (**ksh** and **csh**).

Bash is intended to be a conformant implementation of the Shell and Utilities portion of the IEEE POSIX specification (IEEE Standard 1003.1). **Bash** can be configured to be POSIX-conformant by default.

OPTIONS

All of the single-character shell options documented in the description of the **set** builtin command can be used as options when the shell is invoked. In addition, **bash** interprets the following options when it is invoked:

1 Introduction

1.1 What is Bash?

Bash is the shell, or command language interpreter, for the GNU operating system. The name is an acronym for the '`Bourne-Again SHell`', a pun on Stephen Bourne, the author of the direct ancestor of the current Unix shell `sh`, which appeared in the Seventh Edition Bell Labs Research version of Unix.

Bash is largely compatible with `sh` and incorporates useful features from the Korn shell `ksh` and the C shell `csh`. It is intended to be a conformant implementation of the IEEE POSIX Shell and Tools portion of the IEEE POSIX specification (IEEE Standard 1003.1). It offers functional improvements over `sh` for both interactive and programming use.

While the GNU operating system provides other shells, including a version of `csh`, Bash is the default shell. Like other GNU software, Bash is quite portable. It currently runs on nearly every version of Unix and a few other operating systems − independently-supported ports exist for MS-DOS, OS/2, and Windows platforms.

Bash nie jest zwykłym językiem programowania!

Podstawowe zadanie: uruchamianie programów

1. Wczytaj z linii poleceń nazwę programu i argumenty wywołania.
2. Zażądaj od jądra uruchomienia programu z podanymi argumentami.
3. Odbierz od jądra kod powrotu.
4. Powróć do punktu 1.

Dodatkowe możliwości

- Udogodnienia w przygotowywaniu poleceń do wykonania:
  - Mechanizmy makropodstawień.
  - Wyszukiwanie plików w wielu katalogach (`PATH`, `MANPATH` itp.).
- Prosty interpretowany język programowania pozwalający na podejmowanie decyzji warunkowych i iterację.

Idiosynkrazje Basha

- Spacje w definicjach zmiennych:

  ```
  LOGO="penguin"
  LOGO = "penguin"
  ```

  Pierwszy wiersz, to deklaracja zmiennej `LOGO`, drugi to wywołanie programu `LOGO` z parametrami `=` oraz `penguin`.
- Opcjonalne cudzysłowy i jawny symbol dereferencji:

  ```
  LOGO
  $LOGO
  ```

  Pierwszy wiersz, to napis `LOGO`, drugi — dereferencja zmiennej `LOGO` rozwijająca się do napisu `penguin`.
- Czemu tak? Bo w powłoce systemowej tak trzeba!

Działanie interpretera basha

Wykonuje w pętli poniższe czynności aż osiągnie koniec strumienia wejściowego, wykona instrukcję `exit`, natrafi na błąd itp.

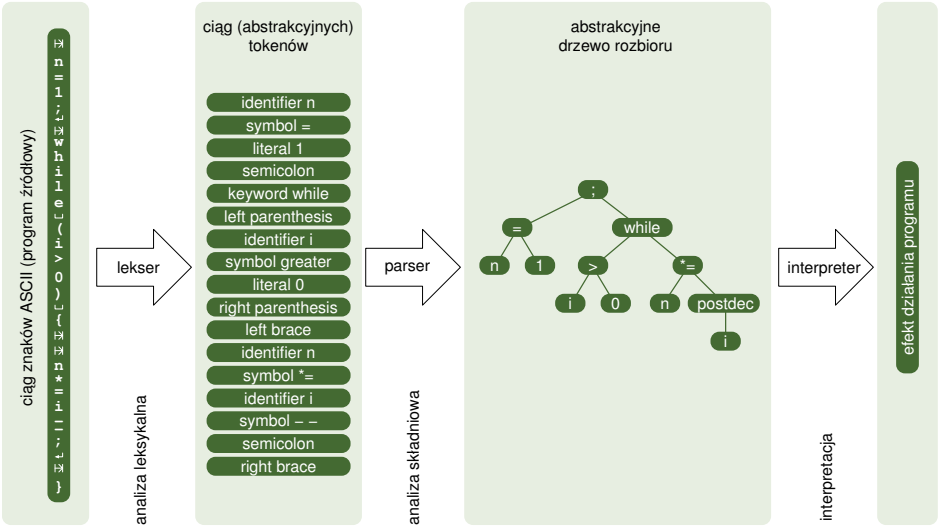
Parsowanie

- Czyta tekst z konsoli lub pliku.
- Dzieli ciągi znaków na tokeny.
- Parsuje ciągi tokenów dzieląc je na instrukcje proste i złożone.
- Zatrzymuje się po przeczytaniu jednej kompletnej instrukcji.

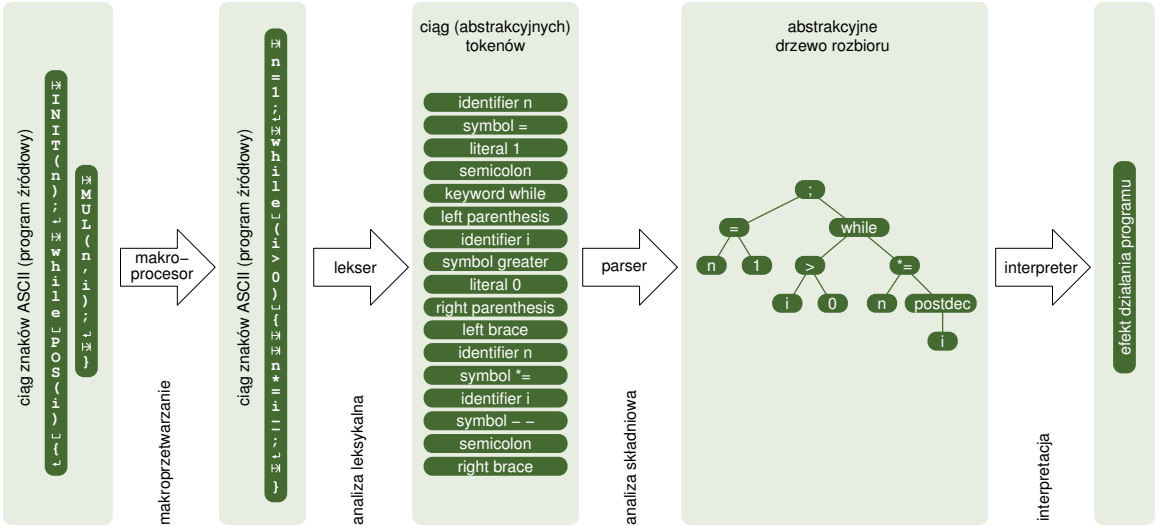
Interpretacja

- Wykonuje cyklicznie poniższe czynności.
- Zgodnie z logiką programu wybiera kolejną instrukcję do wykonania.
- Dokonuje w niej szeregu *rozwinięć*.
- Wybiera i interpretuje umieszczone w niej przekierowania.
- Wykonuje instrukcję.
- Opcjonalnie czeka na jej zakończenie i odczytuje jej kod zakończenia.

Kompilacja i wykonanie języków programowania



Kompilacja i wykonanie języków z makroprocesorem



Preprocesor języka C (nie róbcie tego w domu)

```
#define if if (
#define then ) {
#define else } else {
#define fi }
#define begin {
#define end }

int main(int argc, char* argv[])
begin
  if argc > 1
    then
      printf("Hello %s!\n", argv[1]);
    else
      printf("Hello!\n");
  fi
  return 0;
end
```

cpp -P →

```
int main(int argc, char* argv[])
{
  if ( argc > 1
    ) {
      printf("Hello %s!\n", argv[1]);
    } else {
      printf("Hello!\n");
  }
  return 0;
}
```

Makrogeneratory (makroprocesory)

- Przetwarzają pliki tekstowe, zamieniając pewne ciągi znaków na inne:
  - Rozwijanie makr — w miejsce nazwy makra jest wstawiana treść makrodefinicji. Makra mogą mieć parametry. Zwykle nie są rekurencyjne.
  - Wstawianie w ustalone miejsce zawartości innych plików.
  - Warunkowe wstawianie tekstu.
- Popularne w latach '50-tych jako makroasemblery.
- Specjalizowane wersje są częścią definicji niektórych języków (C, RATFOR). Zwykle transformują tekst programu *przed* rozpoczęciem właściwej kompilacji (*preprocessing*, stąd „preprocesory").
- Nie dokonują analizy składniowej — działają na ciągach znaków. Dlatego są uznawane za „niskopoziomowe" i niebezpieczne (por. `const` i `#define` w ANSI C).
- Popularne implementacje GNU:
  - `gpp` — preprocesor języka C,
  - `M4` — makrogenerator „uniwersalny".
- `gpp` + `make` = PHP dla ubogich — łatwy sposób generowania zbioru statycznych stron w HTML-u.

Kolejność rozwijania w bashu

```
$ MYCMD=echo
$ $MYCMD 1 2 3
```

Kolejność rozwijania w bashu

```
$ MYCMD=echo
$ $MYCMD 1 2 3

$ MYKWD=done
$ for i in 1 2 3; do echo $i; $MYKWD
```

Kolejność rozwijania w bashu

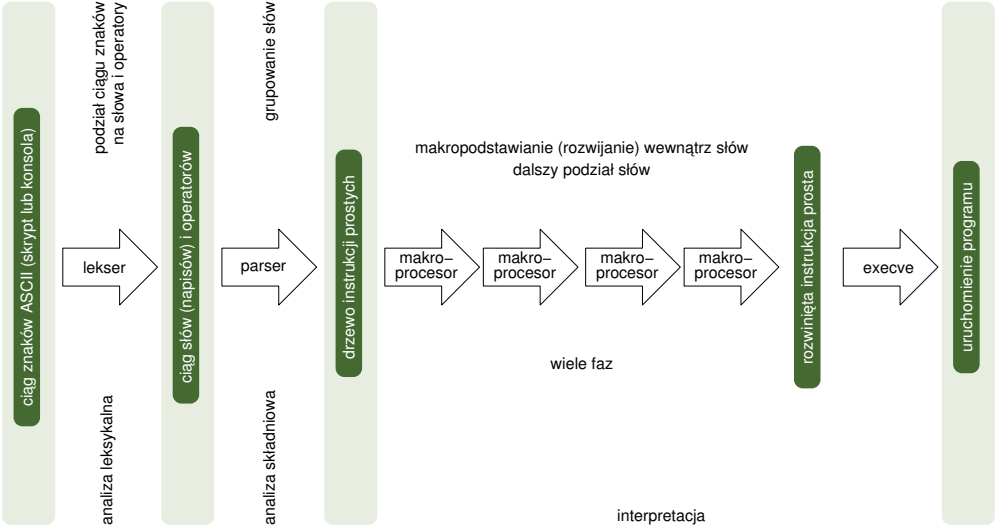
```
$ MYCMD=echo
$ $MYCMD 1 2 3

$ MYKWD=done
$ for i in 1 2 3; do echo $i; $MYKWD

$ $MYKWD
bash: done: command not found
$ done
bash: syntax error near unexpected token ‘done’
$ for i in 1 2 3; do echo $i; !!
```

Kolejność rozwijania w bashu

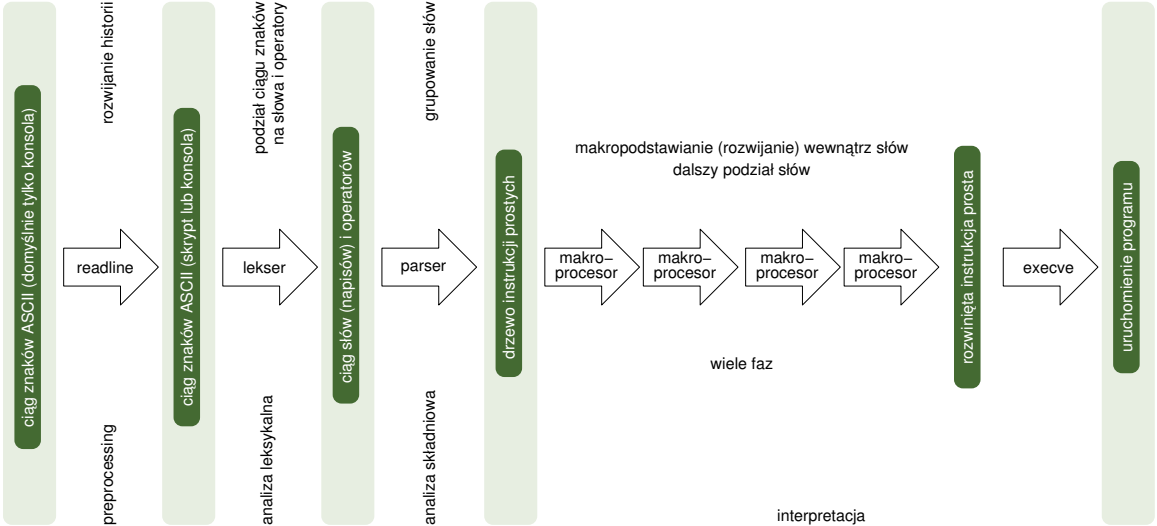
```
$ MYCMD=echo
$ $MYCMD 1 2 3

$ MYKWD=done
$ for i in 1 2 3; do echo $i; $MYKWD

$ $MYKWD
bash: done: command not found
$ done
bash: syntax error near unexpected token ‘done’
$ for i in 1 2 3; do echo $i; !!
```

- Rozwijanie historii odbywa się *przed* kompilacją (to jest *preprocessing*).
- Pozostałe makropodstawienia są wykonywane podczas *interpretacji* skryptu w słowach wykonywanej instrukcji prostej.

W bashu makrogeneracja odbywa się podczas interpretacji



Tylko `readline` dokonuje preprocesingu



Backend basha

Podstawowym efektem interpretacji skryptu bashowego jest wywoływanie funkcji jądra Linuksa (API w pliku nagłówkowym `unistd.h`):

```
pid_t fork(void);
int execve(const char *filename, char *const argv[],
           char *const envp[]);
```

Przygotowanie środowiska (deskryptory plików itp.) oraz argumentów funkcji `execve` odbywa się wskutek interpretacji *instrukcji prostej* postaci:

[*przypisania-zmiennych*] *nazwa-programu* [*argumenty* ...]

Nierozwinięte instrukcje proste są umieszczone w abstrakcyjnym drzewie rozbioru, po którym porusza się interpreter.

Bash posiada ponad 50 *poleceń wbudowanych*. Zamiast wykonywać `execve` na zewnętrznym programie, bash wykonuje je samodzielnie. Por. `busybox`.

Celem pracy basha jest zlecanie uruchamiania programów

Instrukcja prosta `ls *` jest rozwijana do `ls` *`plik1 plik2 plik3...`* i następuje wywołanie

```
execve("/bin/ls", argv, envp);
```

gdzie `argv` jest tablicą argumentów programu, zaś `envp` — tablicą środowiska (napisów postaci *zmienna=wartość*).

W wyniku wykonania `ls *` można czasem otrzymać komunikat o błędzie:

```
bash: Argument list too long
```

Czemu?

Celem pracy basha jest zlecanie uruchamiania programów

Instrukcja prosta `ls *` jest rozwijana do `ls` *`plik1 plik2 plik3...`* i następuje wywołanie

```
execve("/bin/ls", argv, envp);
```

gdzie `argv` jest tablicą argumentów programu, zaś `envp` — tablicą środowiska (napisów postaci *zmienna=wartość*).

W wyniku wykonania `ls *` można czasem otrzymać komunikat o błędzie:

```
bash: Argument list too long
```

Czemu?

- Tablice `argv` i `envp` powinny razem zmieścić się w pamięci o rozmiarze zwracanym przez `getconf ARG_MAX` (u mnie 2097152 = 2 MiB).
- Jeśli nie, to `execve` zwraca błąd `E2BIG` (poprzez `errno`).

To jest ograniczenie nakładane przez konfigurację jądra, a nie przez basha.

Wieloprocesowość w Linuksie

- Jądro uruchamia program `/sbin/init` (PID 1).
- `/sbin/init` uruchamia dalsze procesy, np. `/sbin/getty`, który uruchamia `/bin/login`, który uruchamia `/bin/bash`, który uruchamia dalsze procesy.
- Strona wywołująca:

```
int execve(const char *name, char *const argv[], char *const envp[]);
```

- Strona wywoływana:

```
int main(int argc, char *argv[], char *envp[]);
```

- Strona wywoływana kończy pracę wykonując `return` n w `main`.
  Wartość n (0–255) to *kod powrotu*.
- Strona wywołująca otrzymuje kod powrotu (wykonując `waitpid`).
- Wyrażenia sterujące w instrukcjach warunkowych i pętli w bashu to też instrukcje basha (proste lub złożone).
- Instrukcje warunkowe i pętli podejmują decyzje na podstawie kodu powrotu z wykonania wyrażenia sterującego.
- Uwaga: $n = 0$ — prawda, $n \neq 0$ — fałsz!

Struktura i interpretacja basha

Frontend

- Interaktywny: Biblioteki GNU Readline i GNU History.
- Wsadowy: skrypty.

Kompilacja

- Struktura leksykalna.
- Składnia.

Interpretacja

- Wykonywanie instrukcji złożonych w celu wyboru do wykonania instrukcji prostych.
- Rozwijanie instrukcji prostych.

Backend

- Uruchamianie programów i wykonywanie poleceń wbudowanych.

Biblioteki GNU Readline i GNU History

- Uniwersalny edytor wiersza poleceń.
- Rozbudowana edycja wiersza ze skrótami klawiszowymi w stylu Emacs-a oraz vi.
- Historia wcześniej wprowadzonych wierszy z rozbudowanymi możliwościami wyszukiwania i edycji.
- Bardzo łatwa integracja z dowolnym programem w C, Perlu, Tcl/Tk, Pythonie i innych językach:

```
input = readline(prompt);
```

- Biblioteki współdzielone: `libreadline.so.7.0` i `libhistory.so.7.0`.
- Pakiety w Debianie: `readline-common`, `readline-doc`, `libreadline7`, `libreadline7-dev`, `libreadline7-dbg`.
- Dokumentacja:
  - `readline(3)`, `history(3)`
  - *The Gnu Readline Library*, Brian Fox and Chet Ramey (Texinfo)
  - *The Gnu History Library*, Brian Fox and Chet Ramey (Texinfo)

Readline

Pliki konfiguracyjne

`/etc/inputrc`
`~/.inputrc`

- Ponad 100 komend edycji wiersza.
- Możliwość dowolnego przypisywania konfiguracji klawiszy do komend.
- Dużo komend, które standardowo nie mają przypisanych kombinacji klawiszy.
- Ponad 40 zmiennych konfiguracyjnych.
- Preprocesor plików konfiguracyjnych: warunkowa kompilacja i dołączanie zawartości plików.

Readline w bashu

- Dodatkowe przypisania kombinacji klawiszy.
- Domyślnie działa w trybie interakcyjnym. Opcja `--noediting`.
- Polecenie `set -o emacs` włącza edycję w stylu emacsa, a `set -o vi` — w stylu vi.

Komendy edycji wiersza

Przesuwanie kursora

C-a, <HOME>	beginning-of-line
C-e, <END>	end-of-line
C-f, <RIGHT>	forward-char
C-b, <LEFT>	backward-char
M-f, C/M-<RIGHT>	forward-word
M-b, C/M-<LEFT>	backward-word
C-l	clear-screen

Historia

<Return>, C-j, C-m	accept-line
C-o	oper-and-get-next
C-x C-e	edit-and-exec
M-C-e	shell-expand-line
M-C-y	yank-nth-arg
M-., M-_	yank-last-arg
C-p, <UP>	previous-history
C-n, <DOWN>	next-history
M-<	beginning-of-history
M->	end-of-history
C-r	reverse-search-history
C-s	forward-search-history
M-p	non-inc-rev-search
M-n	non-inc-fwd-search
M-^	history-expand-line

Edycja

C-d	end-of-file
C-d, <DEL>	delete-char
<BSP>	backward-delete-char
C-v, C-q	quoted-insert
C-t	transpose-chars
M-t	transpose-words
M-u	upcase-word
M-l	downcase-word
M-c	capitalize-word

Komendy edycji wiersza (2)

Usuwanie

C-k	kill-line
C-x <BSP>	backward-kill-line
C-u	unix-line-discard
M-d	kill-word
M-<BSP>	backward-kill-word
C-w	unix-word-rubout
M-\	delete-horizontal-space
C-y	yank
M-y	yank-pop

Automatyczne uzupełnianie

<TAB>	complete
M-?	possible-completions
M-*	insert-completions

Dodatkowe uzupełnianie w bashu

M-/	complete-filename
C-x /	possible-filename-compl
M-~	complete-username
C-x ~	possible-username-compl
M-$	complete-variable
C-x $	possible-variable-compl
M-@	complete-hostname
C-x @	possible-hostname-compl
M-!	complete-command
C-x !	possible-command-compl
M-<TAB>	dynamic-complete-history
M-{	complete-into-braces

Makra

C-x (	start-kbd-macro
C-x)	end-kbd-macro
C-x e	call-last-kbd-macro

Argumenty liczbowe komend

M-0, M-1, ..., M-9, M--	digit-arg

Komendy edycji wiersza (3)

Różne

C-x C-r	re-read-init-file
C-g	abort
M-abc...	do-uppercase-version
<ESC>	prefix-meta
C-_, C-x C-u	undo
M-r	revert-line
M-&	tilde-expand
C-@, M-<SPC>	set-mark
C-x C-x	exch-point-and-mark
C-]	character-search
M-C-]	char-search-backward
M-#	insert-comment
M-g	glob-complete-word
C-x *	glob-expand-word
C-x g	glob-list-expansions
C-x C-v	display-shell-version

Edycja w trybie incremental search

C-s	forward search next
C-r	reverse search next
C-r C-r	recall previous search
C-g	abort search
<ESC>, C-j	terminate search
any other	terminate search and
bash command	execute command
e.g., <RETURN>	

Rozwijanie historii

Składania „wyrażeń historycznych”

- Wybieranie wiersza historii: `!`*n*, `!-`*n*, `!!` (= `!-1`), `!`*str*, `!?`*str*[`?`], `^`*str*`^`*str*`^`, `!#`.
- Wybieranie fragmentu wiersza: `:`*n* ($n \geq 0$), `:`*n*`-`*m*, [`:`]`^` (= `:1`), [`:`]`$`, `:`*n*`*` (= `:`*n*`-$`), `:`*n*`-`, [`:`]-*n* (= `:0-`*n*), [`:`]`*`, [`:`]`%`.
- Modyfikatory: `h`, `t`, `r`, `e`, `p`, `q`, `x`, `s/`*old*`/`*new*`/`, `&`, `g`, `G`.

Najczęściej używane odwołania do historii

- `!!` — poprzednie polecenie
- `!$` — ostatnie słowo poprzedniego polecenia (skrót od `!!$`)
- `!:2` — drugi argument poprzedniego polecenia (trzecie słowo)
- `!gcc` — ostatnio wprowadzony wiersz zaczynający się znakami `gcc`

Dobre rady

- Kombinacja klawiszy `M-^` działa jak gpp w C — rozwija historię bez kompilowania wiersza.
- `M-2 M-.` ma ten sam efekt, co wpisanie `!:2` i naciśnięcie `M-^`.
- Znaki `\!` w `PS1` wstawiają numer instrukcji do tekstu zachęty.