Kurs administrowania systemem Linux

Zajęcia nr 2: Powłoka systemowa

Instytut Informatyki Uniwersytetu Wrocławskiego

8 marca 2022

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.

Polecenia basha w pliku — skrypty

- Domyślnie rozwijanie historii jest wyłączone.
- Bash kompiluje i wykonuje tylko jedną instrukcję na raz.
- Wniosek: w skrypcie mogą być błędy składniowe, które nie zostaną wykryte podczas wykonania!
- Wykonywanie skryptu:

  `$ bash` *`plik-z-programem`*

  `$ bash -c '`*`tekst programu`*`'`
- Można nadać plikowi z programem prawa do wykonania:

  `$ chmod a+x` *`plik-z-programem`*

  i uruchamiać poleceniem

  `$` *`plik-z-programem`*

  jak zwykły program.

#! (*hash-bang*, *she-bang*)

Uruchomienie skryptu z prawami do wykonania jako programu

- Pierwszy wiersz skryptu postaci

  `#!`*`nazwa-interpretera argumenty`*

  powoduje wykonanie instrukcji

  `nazwa-interpretera argumenty plik-z-programem`
- Np. jeśli plik wykonywalny `myprog` zawiera wiersz

  `#!/usr/bin/gawk -f`

  to polecenie

  `$ myprog`

  spowoduje wykonanie programu

  `/usr/bin/gawk -f myprog`
- *Konwencja:* jeśli *plik-z-programem* nie zawiera *hash-bang*, to nie powinien mieć prawa do wykonania, a jego nazwa powinna mieć rozszerzenie .sh. Jeśli zawiera *hash-bang*, to powinien mieć prawa do wykonania, a nazwa nie powinna zawierać żadnego rozszerzenia.

Struktura leksykalna basha

- Komentarze zaczynają się znakiem `#` i kończą znakiem nowego wiersza (w trybie interaktywnym można wyłączyć).
- Ciąg `\<newline>` jest usuwany.
- *Biały znak*: spacja lub znak tabulacji.
- *Metaznak*: znak nowego wiersza, `|`, `&`, `;`, `(`, `)`, `<`, `>`.
- *Token*: *słowo* (wymaga oddzielenia białymi znakami) lub *operator* (nie wymaga).
- *Słowo*: dowolny ciąg znaków różnych niż białe i metaznaki.
- *Operator*: token zbudowany z jednego lub więcej metaznaków.

Operatory i słowa kluczowe

- Operatory dzielą się na *sterujące* i *przekierowania*.
- *Operator sterujący*: znak nowego wiersza, `||`, `&&`, `&`, `;`, `;;`, `;&`, `;;&`, `|`, `|&`, `(`, `)`.
- *Operator przekierowania*: `<`, `>`, `<<`, `>>`, `<<<`, `<>`, `&>`, `>&`, `<&`, `>&`, `&>>`.
- *Słowo kluczowe*: `!`, `case`, `coproc`, `do`, `done`, `elif`, `else`, `esac`, `fi`, `for`, `function`, `if`, `in`, `select`, `then`, `until`, `while`, `{`, `}`, `time`, `[[`, `]]`. Słowo kluczowe jest zarezerwowane tylko wtedy, gdy nie jest ujęte w cudzysłowy i jest pierwszym słowem instrukcji prostej lub trzecim słowem instrukcji `case` lub `for`.

Quoting

Quoting pozwala tworzyć pojedyncze tokeny zawierające białe znaki i metaznaki:

- *Backslash* `\`: odbiera specjalne znaczenie następnemu znakowi z wyjątkiem `<newline>`.
- Apostrofy `'...'`: odbierają specjalne znaczenie ciągowi znaków.
  Ciąg nie może zawierać `'`.
- Cudzysłowy `"..."`: odbierają specjalne znaczenie ciągowi znaków z wyjątkiem `` ` ``, `$` i `\` (tylko jeśli następuje po nim `` ` ``, `$`, `\`, `"` lub `<newline>`). Zmieniają znaczenie `$*` i `$@`.
- Znaki sterujące w stylu C: `$'...'`. Specjalne znaczenie: `\a`, `\b`, `\e`, `\E`, `\f`, `\n`, `\r`, `\t`, `\v`, `\\`, `\'`, `\"`, `\`*nnn*, `\x`*HH*, `\u`*HHHH*, `\U`*HHHHHHHH*, `\c`*x*.
- Teksty zależne od wersji językowej: `$"..."`.

Składnia basha

- Parser dzieli ciągi słów na zdania zakończone `;` lub `<newline>`.
- Zdania zaczynające się słowem kluczowym są fragmentami instrukcji złożonych.
- Instrukcje proste: [*przypisanie zmiennej* ...] *nazwa-programu* [*argument* ...]
- Potoki: [`time` [`-p`]] [`!`] *instrukcja*$_1$ [[`|`|`|&`] *instrukcja*$_2$ ...]
- Listy instrukcji: *potok*$_1$ [[`;`|`&`|`&&`|`||`] *potok*$_2$ ...]
- Instrukcje złożone (zawierają listy instrukcji, zmienne, wzorce, wyrażenia arytmetyczne i wyrażenia logiczne).
- Za instrukcją złożoną mogą wystąpić przekierowania (uwaga na jednoznaczność!).

Instrukcje złożone

Niech *instr* oznacza instrukcję, *li* — *listę instrukcji*, *sł* — słowo, *zm* — *zmienną*, *wyr* — wyrażenie arymetyczne, *log* — wyrażenie logiczne, a *wz* — wzorzec.

- Instrukcja grupowania: `{` *li* `}`
- Instrukcja podprocesu: `(` *li* `)`
- Instrukcja koprocesu: `coproc` [*zm*] *instr*
- Instrukcja warunkowa:
  `if` *li* `;` `then` *li* [`;` `elif` *li* `;` `then` *li*...] [`;` `else` *li*] `;` `fi`
- Instrukcja wyboru: `case` *sł* `in` [[(] *wz* [| *wz*] ... `)` *li* [`;;`|`;&`|`;;&`]] ... `esac`
- Instrukcja zapytania: `select` *zm* [`in` *sł*...]`;` `do` *li* `;` `done`
- Instrukcja pętli: [`while` | `until`] *li* `;` `do` *li* `;` `done`
- Instrukcja iteracji: `for` *zm* [`in` [*sł*...] `;`] `do` *li* `;` `done`
- Instrukcja iteracji arytmetycznej: `for ((` *wyr* `;` *wyr* `;` *wyr* `))` `;` `do` *li* `;` `done`

Wyrażenia, funkcje i zmienne

Wyrażenia

- Wyrażenie arytmetyczne: `(( wyr ))`, por. instrukcję `let`
- Wyrażenie logiczne: `[[ log ]]`, por. instrukcje `test` i `[`

Funkcje

- Składnia: [`function`] *zm* [`()`] *instrukcja złożona*
- Musi wystąpić co najmniej jeden z tokenów `function` i `()`
- Funkcje są rekurencyjne
- Wywołanie funkcji jest instrukcją prostą

Zmienne

- Składnia: ciąg liter, cyfr i znaku `_` nie zaczynający się cyfrą lub zmienna specjalna.
- Zmienne specjalne: `*`, `&`, `#`, `?`, `-`, `$`, `!`, `0`, n ($n > 0$), `_`
- Odwołanie do zmiennej („dereferencja”): `$`[`{`]*zmienna*[`}`]

Makropodstawienia w bashu

Na tekście instrukcji prostej wybranej do wykonania wykonuje się w kolejności ciąg makropodstawień:

- rozwinięcia nawiasów wąsatych, np. `file{1,2,3}`, `file{1..10}`,
- rozwinięcia tyldy, np. `~/Downloads/`,
- rozwinięcia zmiennych, np. `$HOME`,
- podstawienia instrukcji, np. `$(cat file.txt)`,
- podstawienia procesów, np. `<(pdftops file.pdf -)`,
- rozwinięcia arytmetyczne, np. `$((N+1))`,
- (powtórny) podział słów,
- rozwinięcia nazw plików, np. `file?-*.txt`.