# Narzędzia Wspierające Programowanie

## Bash – poszerzenie podstaw

Zasiadamy przy terminalu Linuxowym. Sprawdźmy podstawowe cechy i zasoby.

Ilość pamięci: `$ free -h`

Miejsce na dysku: `$ df -H .`

Miejsce na koncie: `$ quota -s -u {MyLogin}`

Nazwa dystrybucji Linuxa: `$ cat /etc/issue`

Ilość wątków CPU: `$ nproc`

*Dodatek*

Adres IP komputera: `$ hostname -I`

Adres DNS komputera: `$ host `hostname -I``

Informacje o CPU: `$ less /proc/cpuinfo` (wyjście: q)

Wypis zawartości katalogu - można go ukierunkować.

Samo `ls` wypisuje tylko nazwy, a `ls -l` ma długi wypis i słabo rozróżnia rodzaje.

Wypis nazw, 1 nazwa na linię: `$ ls -1` (przyda się w obróbce seryjnej)

Wypiszmy tylko ścieżki: `$ ls -d */`

→ Propozycja: `$ ls -ogh --color`

Krótszy wypis, czytelne długości plików, koloryzacja ścieżek, ...
(*komenda jest długa – zaradzimy temu przy konfiguracji bash'a)*

*Dodatek*

Wypis plików ukrytych: `$ ls -a` (pliki o nazwach zaczynających się od . )

Wypis w formie drzewa: `$ tree`

## ⦿ Kopiowanie i przenoszenie

Aby w kopii zachować datę: `$ cp -p` (pomaga zrozumieć chronologię)

Kopiowanie z podścieżkami: `$ cp -r`
Przenoszenie ścieżki `$ mv -f`
Kasowanie ścieżki `$ rm -r`

## ⦿ *Dodatek*

Kopiowanie z paskiem postępu `$ gcp` (przy dużych danych – szacujemy postęp)

## ⦿ Poszukiwanie plików

Aby wyszukać plik(i): `$ find {od/sciezki} -name {plik/i}`
Np.: `$ find . -name mojPlik.\*`

Aby wyszukać ścieżki: `$ find {od/sciezki} -type d -name {ścieżka/i}`

## ⦿ Wyszukiwanie fraz w środku plików

Aby wyszukać frazy: `$ grep {fraza} {plik}`

Często jednak szukamy frazy, nie wiedząc, w którym jest pliku, a np. jest wiele podkatalogów.

→ Propozycja: `$ grep --color -inr --include=*.{C,h} fraza`

gdzie:

| | |
|---|---|
| `-r (recursive)` | szukaj w podkatalogach |
| `-i (ignore case)` | akceptuj każdą wielkość liter |
| `-n (line number)` | wypisz nr linii, gdzie jest fraza |
| `--include={type}` | szukaj wśród plików tego typu |

(*komenda jest długa – zaradzimy temu przy konfiguracji bash'a)*

## ⦿ Linki symboliczne

Jeśli plik ma występować w kilku katalogach lub pod kilkoma nazwami,
to kilka kopii tego samego - marnotrawi zasoby na dysku.
Lepiej utworzyć link symboliczny do oryginału.

Link do pliku, adres względny: `$ ln -s {ścieżka_stąd}/plik_oryginalny .`

Gdy plik jest na ścieżce obok naszej: `$ ln -s ../{ścieżka_obok}/plik_oryginalny .`

Podanie pełnej lokalizacji: `$ ln -s /pelna/ścieżka/do/plik_oryginalny .`

Wypis linku i jego źródła: `$ ls -l {link_symboliczny}`

## ⦿ *Dodatki*

Wypis tylko ścieżek w katalogu: `$ ls -l | grep ^d`

Wypis tylko linków w katalogu: `$ ls -l | grep ^l`

Jak szybciej pisać w terminalu dzięki skrótom klawiszowym:

| | |
|---|---|
| `[Ctrl Shift C/V]` | copy/paste w terminalu |
| `[Ctrl ←/→]` | w lewo/prawo o słowo |
| `[Ctrl A]` | skok na początek linii |
| `[Ctrl E]` | skok na koniec linii |
| `[Ctrl U]` | kasuj od kursora w lewo |
| `[Ctrl K]` | kasuj od kursora w prawo |
| `[Ctrl D]` | kasuj znak pod kursorem |
| `[Ctrl W]` | kasuj od kursora do początku słowa |
| `[Alt D]` | kasuj od kursora do końca słowa |
| `[Ctrl T]` | zamień literę pod kursorem z poprzednią |
| `[Ctrl R]` | podpowiedzi z poprzednich komend |

Szersze zestawienie skrótów klawiszowych i pomocniczych komend np. tu: Link

## ⦿ Czytniki plików tekstowych

Często chcemy na szybko przejrzeć plik (np. z danymi – wtedy plik może być b. długi).
Wciąganie długiego pliku do edytora długo trwa. ⇒ Warto znać szybkie czytniki tekstowe.
Również, czytnik zabezpiecza przed nieumyślnym nadpisaniem.

- `less {plik}` : Szybki (nawet b. długi plik otworzy się w moment)
  Brak koloryzacji składni kodów

| | |
|---|---|
| `less -N {plik}` | numeruj linie |
| `PgDn , PgUp` | skok do następnej / poprzedniej strony |
| `g, G` | skok na koniec / początek pliku |
| `/fraza` | szukaj wystąpienia frazy |
| `n , N` | szukaj następne/poprzednie wystąpienie |
| `q` | wyjście |

- `view {plik}` : Czytnik będący obcięciem edytora vim do trybu read-only
  Szybki (długie pliki otwiera w moment)
  Koloryzacja składni kodów (ułatwia lekturę)

| | |
|---|---|
| `PgDn , PgUp` | skok do następnej / poprzedniej strony |
| `[Shift g] , gg` | skok na koniec / początek pliku |
| `/fraza` | szukaj wystąpienia frazy |
| `n , N` | szukaj następne/poprzednie wystąpienie |
| `:set nu` | numeruj linie |
| `:q` | wyjście |

## ⦿ *Dodatek*

- `nano {plik}` : Edytor prosty w obsłudze, Koloryzacja składni kodów.
  Możliwość edycji kilku plików na raz

| | |
|---|---|
| `[Ctrl W] fraza` | szukaj frazy |
| `[Ctrl+W]` | szukaj kolejnego wystąpienia |
| `[Alt+B]` | tryb szukania wstecz |
| `[Ctrl W] [Ctrl T] {nr}` | skocz do linii nr. |
| `[Ctrl K]` | skasuj aktualną linię, ale zapamiętaj ją w schowku |
| `[Ctrl U]` | wstaw tu linię ze schowka |
| `[Alt Shift #]` | numeruj linie |
| `[Ctrl R]` | otwórz nowy plik w nowym „buforze” |
| `[Alt > ]  [Alt < ]` | przeskok do edycji następnego / poprzedniego pliku |

- Typowy kompresor plików — `gzip {pliki}` → `{pliki}.gz`

  i dekompresja: `gunzip {pliki}.gz` lub `gzip -d {pliki}.gz`

- Pakowacz plików: `tar {archiwum} {plik1} {plik2} …` {również wildcards}

  Typowe pakowanie: `tar czvf {archiwum.tgz} {pliki źródłowe}`

  gdzie:

  | | | |
  |---|---|---|
  | `c` | (create) | utwórz archiwum |
  | `z` | (zip) | skompresuj |
  | `f` | (file) | dotyczy plików |

  Wypis zawartości archiwum `tar tf {archiwum}`

  Typowe rozpakowywanie `tar xzvf {archiwum.tgz}`

  gdzie:

  | | | |
  |---|---|---|
  | `x` | (extract) | wypakuj |

⊕ *Dodatek:* Przy Big Data :

- Wielowątkowy kompresor: `pigz {pliki}` lub `pigz -n {L. wątków} {pliki}`
- Połączenie tar i pigz `tar -c -I pigz -f archive.tgz {pliki_zrodlowe}`

## Zdalne kopiowanie

- Ściągnięcie pliku z internetu `wget {url}`

  gdy plików w folderze jest więcej: `wget --recursive --no-parent {url/folder}`

  aby uciąć do n-tego subfoldera: `wget --recursive --no-parent --cut-dirs=n {..}`

- Kopia z/na zdalny komputer `scp` lub `rsync`

→ Propozycja: `rsync -avzP {źródło} {login@node:ścieżka/od/home}`

gdzie:

| | | |
|---|---|---|
| `a` | (archive) | zachowa właściwości plików |
| `z` | (zip) | kompresja w locie (szybciej) |
| `P` | (partial+progress) | pokazuje postęp kopiowania<br>+ przy przerwaniu pozwala wznowić |

`rsync` sprawdza, czy części paczki nie ma już w miejscu docelowym.
Kopiuje, gdy ich nie ma lub gdy plik docelowy jest starszy (uzysk czasu, przydatne do backupu)

## Jak obsługiwać procesy.

- Podejrzyjmy je w monitorze: `$ top lub htop` (wyjście: q)
- i wypiszmy w terminalu: `$ ps`

(Skrót PID oznacza numer procesu).
Widzimy jednak tylko procesy w naszej sesji.
Aby zobaczyć wszystkie (nasze), wpiszmy:

`$ ps -u {login}`

Warto też spojrzeć na ich hierarchię: `$ ps -u {login} --forest`

Widzimy, że każdy proces jest „podpięty”
pod poprzednika, który go wywołał.

- Skasujmy proces, podając jego PID: `$ kill {PID}`

kasowanie na silniejszym priorytecie: `$ kill -9 {PID}`

*Uwaga*: skasowanie procesu-rodzica wyśle sygnał kasowania do procesów-dzieci.
Zatem, skasowanie terminala (lub sesji ssh) wyłączy wszystkie programy pod to podpięte.

## *Dodatek*

- Zamknięcie okna graficznego: `$ xkill` (a teraz kliknij myszką w okno)
- Wypiszmy „rodziców” procesów (PPID): `$ ps -f`
- Wypiszmy nr/y procesów, gdy znamy nazwę: `$ pidof {nazwa}`

## ⦿ Procesy c.d.: front, tło, pauzowanie

Włączmy w sesji program: `$ sleep 200`

Tryb, w którym działa teraz proces, nazywa się foreground (front):
proces ma kontakt z terminalem, może tam kierować napisy.

Zapauzujmy teraz nasz program: `$ [Ctrl Z]`

Wypiszmy listę zadań włączonych w naszej sesji:

```
$ jobs
[1]+  Stopped                 sleep 200
```

Nasze zadanie ma na tej liście nr. 1. Wznówmy je,
ale niech przejdzie do trybu background (tło): `$ bg %1`

⇒ nie blokuje terminala, ale działa (sprawdź: `jobs`).

Wysuńmy je teraz z tła na front: `$ fg %1`

Teraz przerwijmy nasz proces (zakończmy go): `[Ctrl C]`

Włączmy teraz program z oknem graficznym: `$ gedit`

## ▸ Operator & (ampersand)

Włączenie grafiki blokuje terminal aż do jej wyłączenia (lub zapauzowania przez [Ctrl Z]).
Linux ma operator & (ampersand) , który od razu wpuszcza program do tła.

Spróbujmy: `$ gedit &`

⇒ mamy kontakt i z terminalem, i z edytorem.

Nb.: procesy w tle są wciąż podpięte pod sesję: `$ ps -u {login} --forest`

## ⦿ Pliki konfiguracyjne basha:

`~/.bashrc` wywoływany podczas logowania (ssh lub otwarcie desktopu)
`~/.bash_login` wywoływany np. przy nowym terminalu (w ramach tego samego logowania)

Przykładowy, prosty plik `~/.bash_login`

```
#!/bin/bash
echo Running ~/.bash_login
alias ls="ls --color"
alias fn="find . -name"
alias cgrep="grep --color -inr --include=*.{C,cpp,cxx,c,h,hpp}"
alias rsync="rsync -avzP"
```

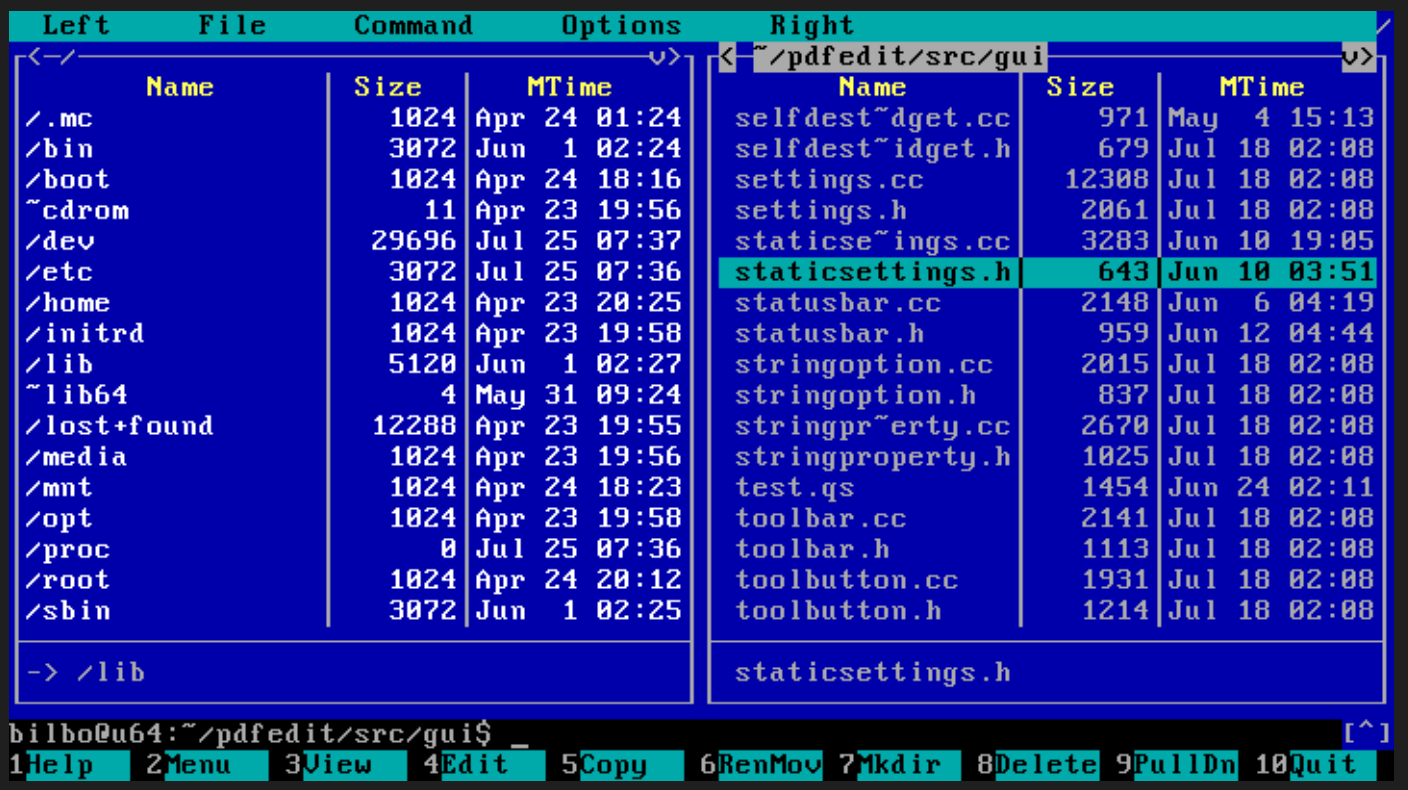
Nadajmy mu prawo do wykonania: `$ chmod 755 ~/.bash_login`

Dla pewności, że się uruchomi, do .bashrc dodajmy na koniec:

```
. ~/.bash_login
```

## ⦿ *Dodatek*:

- `mc` (Midnight Commander) - tekstowa przeglądarka plików




[Tab] Przeskok pomiędzy oknami
[F3] Czytnik pliku
[F8] Skasuj plik (ratuje, gdy plik ma w nazwie trudne znaki kontrolne)
[F10] Wyjście