Przegląd wybranych komend Shella

Paweł Józiak

Wydział Matematyki i Informatyki Politechniki Warszawskiej

25 X 2021

Plan zajęć

Plan zajęć

Podręcznik (*manual*) – *man*

- Informacje o dowolnej komendzie, nazwie pliku systemowego, sys-callu, funkcji C bądź ogólnych tematach około-systemowych, są uzyskiwalne za pomocą komendy `man`, która zbiera, indeksuje, i wyświetla wpisy ich dotyczące.
  ✍ *Możemy wyświetlić informacje o komendzie `man`*

Informacje o komendzie *man*

```
$ man man
```

- Przeglądanie strony podręcznika:
  - `q` – wyjście,
  - <SPACE> – następna strona,
  - *n*`G` – przejście do linii numer *n*,
  - `G` – przejście do ostatniej linii,
  - */wzorzec* – przeszukiwanie w przód pod kątem wystąpienia *wzorca*, znaki specjalne we wzorcu muszą być wyescape'owane,
  - `n`/`N` – przejście do kolejnego/poprzedniego wystąpienia.

Struktura stron i sekcji manuala

- Większość wpisów do stron manuala przestrzega poniższego formatu:
  - NAZWA,
  - SKŁADNIA,
  - OPIS,
  - OPCJE,
  - STATUS WYJŚCIA,
  - PLIKI (pliki powiązane),
  - ZOBACZ TAKŻE (nazwy sekcji i numery stron wpisów do manuala),
  - AUTORZY i RAPORTOWANIE BŁĘDÓW
- Manual jest zorganizowany w 8 sekcji, ponumerowanych jak niżej (w niektórych Sys-V sekcje 4, 5 oraz 7 mogą być wymienione pozycjami):
  1. Komendy użytkownika,
  2. Zapytania systemowe,
  3. Funkcje bibliotek języka C,
  4. Sprzęty oraz pliki specjalne,
  5. Formaty plików oraz konwencje,
  6. Gry i in.
  7. Rozmaite,
  8. Narzędzia administracyjne systemu oraz Daemony.

Struktura i używanie manuala

- Dodatkowe sekcje POSIXowe bywają zdefiniowane dla niektórych komend. Odpowiadają temu samemu wzorcowi, lecz z sufiksem numeru strony `P` (na przykład `1P` oznaczać będzie wpis do POSIXowej strony o komendach użytkownika).
- `man` bez numeru strony wyświetla pierwsze (j/w) dopasowanie.
- Możemy określić numer sekcji, albo wywołać `man -a`, by wyświetlić wszystkie dopasowania, jedno po drugim.
  ✍ *Proszę zwrócić uwagę, że numery sekcji nie są poprzedzone myślnikiem (`-`), jak inne opcje:*

Wskazywanie sekcji manuala

```
$ man kill # pierwsze trafienie
$ man 1 kill # strona manuala o komendzie ‘kill’
$ man 2 kill # strona manuala o zapytaniu systemowym ‘kill’
$ man -a kill # wszystkie wpisy manuala dotyczące ‘kill’
```

Wpisy manuala dla plików i funkcji

- Dla funkcji specyfikujemy jedynie ich nazwę, nie wywołanie (zatem bez nawiasów).
- Wskazując na sprzęt bądź plik specjalny, używamy tylko nazwy sprzętu/pliku, nie jego pełną ścieżkę; należy wskazać adekwatną sekcję.
- Należy omijać identyfikatory kopii pliku/sprzętu.

Wpisy manuala dla plików, sprzętów i funkcji

```
$ man 3 printf # nie zaś: man printf()
$ man 5 passwd # nie zaś: man /etc/passwd
$ man 4 sd # nie zaś: man /dev/sda2
```

Plan zajęć

Bezpieczna powłoka (*Secure SHell*) – *ssh*

- Pozwala na połączenie do zdalnego *TTY* (bądź aplikacji emulującej *TTY*) za pomocą sieci TCP/IP.
- Zapewnia bezpieczny kanał komunikacyjny.
- Rozłączenie bądź zamknięcie terminala skutkuje zakończeniem wszystkich procesów odpalonych za pomocą danej sesji SSH.
- Dostępne dla użytkowników Windowsa za pomocą aplikacji `PuTTY` (`https://putty.org/`):
  - działa pod systemami 32-bitowymi oraz 64-bitowymi,
  - pojedynczy plik *exe*, bez konieczności instalacji,
  - licencja FLOSS (otwarte oprogramowanie).
- Dostępny dla użytkowników Unixa za pomocą komendy `ssh` .

Łączenie ze zdalną maszyną za pomocą *ssh*

```
$ ssh [uzytkownik@]host[.domena]
```

- `host` może być nazwą maszyny bądź jej adresem ip; `domena` może być pominięta, jeśli jest ta sama, co maszyny z której następuje połączenie; ostatnia uwaga aplikuje się również do pola `uzytkownik`.

Narzędzie GNU *screen*

- `screen` uruchamia nową instancję powłoki, niepowiązaną z żadnym terminalem. Można się z takiej sesji wyłączyć (a nawet wylogować), zostawiając samą sesję aktywną. Można wówczas włączyć inną sesję.
- `screen` ma zwłaszcza zastosowanie w przypadku procesów długotrwałych, typu: upgrade/backup systemu, długotrwałe obliczenia, aplikacje w trybie Idle/Rest.
- Sesje mają identyfikator numeryczny, nadawany automatycznie, oraz literalny, kontrolowalny przez użytkownika (bądź domyślny).Jednego z nich wystarcza, by nawiązać ponowne połączenie z sesją `screen` .
- Sposób pracy:
  - `screen [-S nazwasesji]` pozwala wystartować sesję,
  - kombinacja <CTRL> + <A> <D> odłączenie od bieżącej sesji,
  - `screen -r [-D] [nazwasesji]` dołączenie do już istniejącej sesji (nazwa nie jest konieczna, jeśli użytkownik wystartował tylko jedną sesję); `-D` odłącza uprzednio sesję w Powłoce, w której jest ona aktualnie dołączona.
  - `exit` zamyka bieżącą sesję (albo <CTRL> + <D>).

Plan zajęć

Czas i data: *date*, *cal*

- `date` wyświetla bądź ustawia datę i czas. Implementacje `date` różnią się interfejsem, zajrzyj do dokumentacji w swojej dystrybucji, by odnaleźć szczegóły. `date` systemu GNU będzie omówiona przy okazji *natychmiastowego wywołania komendy* na kolejnej prezentacji. UWAGA: Dla wyświetlania bieżącego czasu używaj komendy `date` . Istnieje komenda `time` , której funkcjonalności odpowiadają raczej stoperowi niż zegarkowi.
- komenda `cal` pozwala wyświetlić kalendarz. Domyślnie bieżący miesiąc, a wraz z opcją `-y` wyświetlony będzie cały rok. Pozostałe opcje zależą od implementacji/dystrybucji.

Informacje systemowe: *uname*

- `uname` wyświetla nazwę systemu (na przykład: *Linux*). Ponadto istnieje szereg opcji, które wzbogacają informacje o systemie:
  - `-m` architektura systemu (procesora),
  - `-n` nazwę komputera,
  - `-p` procesor,
  - `-r` jądro
- `uname -a` wszystkie informacje zbierane przez `uname` (w dość kryptycznym formatowaniu).
  ✍ *Przykładowe wywołanie `uname -a` dla systemu Linux:*

Informacje *uname*

```
$ uname -a
 Linux ssh 5.14.8-arch1-1 #1 SMP PREEMPT Sun, 26 Sep 2021
19:36:15 +0000 x86_64 GNU/Linux
```

Informacje sprzętowe (Linux)

- Poniższe komendy powinny być dostępne w systemie:
  - `lscpu` ,
  - `lspci` ,
  - `lsusb` ,

  Opcja `-v` pozwoli wyświetlić bardziej szczegółowe informacje dla ostatnich 2 komend.
- Komenda `free` pozwala wyświetlać zajętość pamięci RAM.
- Nieużywana pamięć RAM może być wykorzystana jako bufor/cache dla zwiększenia wydajności systemu. Gdy zajdzie konieczność zwiększenia faktycznego użycia RAMu, może być od razu zwolniona.
- Komenda `df` pozwala na wyświetlenie zajętości każdego zasobu dyskowego zamontowanego na maszynie.
- W niektórych systemach dostępna jest opcja `-h` , która wyświetla liczby w sposób czytelny dla człowieka (w kilo-, mega-, giga- bądź terabajtach, zależnie od rzędu wielkości) – dotyczy zarówno `free` jak i `df` .

Użytkownicy: *id, who, w*

- Komenda `id` *`username`* wyświetla informacje o użytkowniku oraz grupach, zarówno w postaci numerycznej, jak i czytelnej dla użytkownika. Domyślnym użytkownikiem jest ten, który wywołuje komendę.
- Komendy `whoami` oraz `groups` są zdezaktualizowane: ich funkcjonalności są duplikowane przez `id -un` oraz `id -Gn`
- `who` wyświetla informacje o użytkownikach zalogowanych w systemie, `w` wyświetla podobną informację wzbogaconą o informację o aktualnie wywoływanej komendzie.
- `write` *`username`* `[`*`terminal`*`]` wysyła wiadomość do użytkownika. Określenie terminala jest konieczne tylko jeśli użytkownik jest zalogowany więcej niż raz.
- `mesg [`*`y/n`*`]` włącza/wyłącza możliwość obsługi wiadomości przychodzących (cf. `write`). Bez żadnych opcji wyświetla aktualny stan tej flagi.

Plan zajęć

echo i *cat*

- `echo` drukuje na standardowe wyjście tekst w formacie interpretowanym przez powłokę. Warto go używać do testowania formatowania komend zawierających znaki specjalne.
- `cat` wyświetla skonkatenowany (ang. con*cat*enated) kontekt plików na standardowe wyjście (domyślnie: na ekran). Z opcją `-n` wyświetla dodatkowo numer linii przed linią.

Wyświetlanie plików 1: *more* i *less*

- `more` wyświetla pliki w trybie paginacji (jeden ekran tekstu naraz). Klawisz <SPACE> pozwa na przewijanie stron. Wsteczne przewijanie nie jest możliwe.
- `less` jest najbardziej zaawansowanym narzędziem do przeglądania treści plików. Jest domyślnie używany przez `man` do wyświetlania manuali, więc te same reguły nawigacji mają tu zastosowanie. `less` jest szczególnie przydatny do przeglądania wielkich plików (na przykład: plików z logami).

Wyświetlanie plików 2: *head* i *tail*

- `head` wyświetla pierwszą część pliku (domyślnie: 10 linii). Opcja `-n` pozwala na wyspecyfikowanie liczby linii, które będą wyświetlane.
- `tail` działa podobnie do `head`, lecz wyświetla ostatnie linie. Opcja `-f` pozwala na niezamykanie strumienia wejściowego, oraz drukowanie nowych linii, jeśli się pojawią. Może być przydatny do śledzenia plików z logami w trakcie ich tworzenia.
  ✍ *Domyślnym plikiem z logami jest `/var/log/messages`. Zwykli użytkownicy zazwyczaj nie mają prawa go czytać.*

Przeglądanie domyślnego pliku logów *syslog-ng*

```
# tail -f /var/log/messages
```

Proste edytory tekstu

- Dla podstawowych metod edycji pliku tekstowego, można użyć jednego z następujących podstawowych narzędzi (nie wszystkie muszą być dostępne na danej dystrybucji GNU/Linux czy BSD)
  - `nano` (bądź `pico`) – edytor tekstu ekosystemu GNU, domyślny dla większości dystrybucji Linux. Lista podstawowych komend jest zaprezentowana w dolnej części ekranu. Symbol ^ reprezentuje klawisz <CTRL>.
  - `joe` – prosty edytor podobny do `nano`. Panel pomocy jest również dostępny, pasek stanu wyświetla informację *CTRL-K H for help*.
  - `ee` (*Easy Editor*) – podstawowy edytor systemów BSD.
- Zmienna środowiskowa `EDITOR` zawiera iformację o domyślnym edytorze tekstu w systemie.

Plan zajęć

Nawigacja w drzewie katalogów cz. 1: *pwd, ls*

- `pwd` (*Path to Working Directory*) wyświetla ścieżkę do bieżącego (roboczego) katalogu.
- `ls` listuje zawartość katalogu posortowaną alfabetycznie po nazwie. Podanie nazwy pliku zamiast katalogu jest możliwe – wyświetla tylko informacje dotyczące wskazanego pliku. Brak argumentu jest traktowany jako bieżący katalog. Popularne opcje to:
  - `-a` uwzględnij wpisy zaczynające się na ‘ . ’ (pliki ukryte),
  - `-A` jak `-a` , lecz bez katalogów `.` oraz `..` ,
  - `-l` szczegółowe informacje o zasobach (prawa dostępu, właściciele, grupy, wielkość, czas ostatniej modyfikacji)
  - `-h` wyświetlanie wielkości w sposób czytelny dla człowieka (tj. w B, kB, MB, GB, TB, w zależności od rozmiaru, tak, aby wyświetlić liczbę ¡ 1024),
  - `-R` rekursywne wyświetlanie zawartości
  - opcje sortowania wyników:
    - -r odwrócony porządek,
    - -S względem wielkości,
    - -t względem czasu ostatniej modyfikacji,
    - -X względem rozszerzenia.

Nawigacja w drzewie katalogów cz. 2: *cd*

- `cd` (*Change Directory*) – zmienia katalog roboczy. Dopuszczalne są ścieżki relatywne. Brak argumentu interpretowany jest jako powrót do `$HOME`. Powrót do ostatnio odwiedzanego katalogu: `cd -` .
  ✍ *Załóżmy, że katalogiem domowym jest `/home/smithj` , i właśnie przeszliśmy do katalogu `/etc` . Poniższe komendy są równowazne:*

Zmiana katalogu roboczego

```
$ cd /home/smithj
$ cd
$ cd $HOME
$ cd ~
$ cd ../home/smithj
$ cd -
```

Tworzenie nowych plików i katalogów: *touch*, *mkdir*

- `touch` aktualizuje czas ostatniego dostępu/modyfikacji pliku. Jeśli plik nie istnieje, nowy pusty jest tworzony.
- `mkdir` tworzy katalog. `mkdir -p` tworzy sekwencję katalogów (tych, które są konieczne – już istniejących nie rusza).
  ✍ *Można zauważyć, że `mkdir -p` nie zwraca błędu jeśli katalog już istnieje:*

Tworzenie wielopoziomowej struktury katalogów

```
$ mkdir sub1/sub2/sub3
  mkdir error: No such file or directory!
$ mkdir -p sub1/sub2/sub3
$ mkdir sub1/sub2/sub3
  mkdir error: File exists!
$ mkdir -p sub1/sub2/sub3
```

Kopiowanie, przenoszenie i zmiana nazwy – *cp, mv*.

- `cp` kopiuje pliki, `mv` – przenosi/zmienia nazwę

Syntaks *cp* i *mv*

```
$ cp source destination
$ mv source destination
```

- Aby kopiować katalogi wraz z zawartością, opcja `-r` bądź `-R` (rekursywnie) musi być włączona.
- Miejsce docelowe (*destination*) może oznaczać nową nazwę pliku albo nową lokalizację pliku.
  ✍*W pierwszym przykładzie* `plik2` *reprezentuje nową nazwę. W drugim –* `folder` *reprezentuje nową lokalizację*

Kopiowanie pliku

```
$ cp plik plik2
$ cp plik folder
```

Linki symboliczne i twarde – *ln*

- Istnieją dwa typy linków:
  - Dwa lub więcej obiektów systemu plików, które wskazują na ten sam zakres fizycznej pamięci – *twarde linki*
  - Obiekt systemu plików, który wskazuje na inny obiekt systemu plików – wzorem Windowsowych *skrótów* – *symboliczne linki* (albo: *symlinki*)
- Symlinki nie są automatycznie aktualizowane przy zmianie ścieżki/nazwy pliku, na który wskazują
- Twarde linki muszą odnosić się do obiektu tym samym systemie plików, zaś symlinki mogą wskazywać na obiekty w innych systemach plików.
- Generalnie powinno się używać jedynie symlinków.
- `ln` tworzy twarde linki, zaś `ln -s` tworzy linki symboliczne. Ich syntaks wygląda następująco:

Syntaks *ln*

```
$ ln -s cel_linku nazwa_linku
```

Usuwanie plików i katalogów – *rm*

- `rm` usuwa pliki/katalogi.
- Jeśli dane są twardo zalinkowane (por. poprzedni slajd) kilkoma linkami, `rm` usuwa pojedynczy wpis w tabeli systemu plików.
- Usuwanie rekursywne (katalogu wraz z zawartością) jest możliwe dzięki użyciu opcji `-r` bądź `-R` (uprzednio używano komendy `rmdir` , ale ona usuwała jedynie puste katalogi).
- Inne przydatne opcje to m.in.:
  - `-i` interaktywne usuwanie: pytanie o potwierdzenie przy każdym pliku, który miałby być usunięty,
  - `-f` ignorowanie uwag usuwania – automatyczne odpowiadanie *yes* na pytania, czy na pewno.

Synchronizacja – *rsync*

- `rsync` synchronizuje katalogi pomiędzy dwiema lokalizacjami: źródłową i docelową. Pliki w katalogu źródłowym pozostają nietknięte.
- `rsync` jest w stanie synchronizować dane pomiędzy zasobami dyskowymi na różnych maszynach.

Typowe użycie *rsync*

```
$ rsync -avz --delete ~/sourcedir
me@otherhost:/home/me/destdir
```

- Powyższe opcje tłumaczą się następująco:
  - `-a` pozwala zachować podstawowe atrybuty plików (uprawnienia etc),
  - `-v` (*verbose*) rozbudowane logi będą pisane na standardowe wyjście,
  - `-z` dla zmniejszenia zużycia protokołu transferu danych, dane są kompresowane przed wysłaniem, i dekompresowane przed zapisem w docelowym miejscu (zwiększa użycie CPU na obu końcach),
  - `--delete` usuwa nadmiarowe pliki w katalogu docelowym.
- Opcja `-n` (albo w syntaksie GNU: `--dry-run`) pozwala odpalić synchronizację w trybie testowym: pojawiają się jedynie komunikaty o tym, co będzie zsynchronizowane.

Porównywanie plików – *diff*

- `diff` porównuje pliki linia po linii. `diff -y` wyświetla różnice w treści plików w dwukolumnowym widoku, obok siebie. Linie opatrzone symbolami $<$, $>$ i $|$ oznaczają linie, w których: linii brakuje w pierwszym pliku, linii brakuje w drugim pliku, linie, które są różne w obu plikach, odpowiednio.
- `diff` (bez opcji) wyświetla modyfikacje konieczne do tego, by zmienić plik pierwszy w drugi. Ten rezultat może być zapisany do pliku różnicowego, który może być użyty przez komendę `patch` (por. nast. slajd).

  ✍ *`plik.patch` zawiera informacje o tym, jak `stary.plik` powinien być zmieniony, by otrzymać `nowy.plik`*

Generowanie pliku różnicowego

```
$ diff stary.plik nowy.plik > plik.patch
```

Aplikowanie plików różnicowych – *patch*

- `patch` wykonuje zmiany na pliku, zgodnie z instrukcją przekazaną przez plik różnicowy wygenerowany przez `diff`

Aplikowanie zmian wynikających z pliku różnicowego

```
$ patch stary.plik plik.patch
```

Cofanie zmian wynikających z pliku różnicowego

```
$ patch -R nowy.plik plik.patch
```

Plan zajęć

Kompresja – *gzip* oraz *bzip2*

Kompresowanie plików

```
$ gzip somefile1
$ bzip2 somefile2
$ xz somefile3
```

- `gzip` , `bzip2` oraz `xz` kompresują pliki oraz dodają rozszerzenia `.gz` , `.bz2` oraz `.xz` , odpowiednio.
- Tym komendom można podać więcej niż jeden plik do argumentu. Wówczas każdy plik kompresowany jest osobno.

Dekompresja

```
$ gunzip somefile1.gzip
$ bunzip2 somefile2.bz2
$ unxz somefile3.xz
```

- `gunzip` , `bunzip2` oraz `unxz` dekompresują pliki oraz usuwają rozszerzenia (`.gz` , `.bz2` czy `.xz`)

tar – Tape ARchiver

- `tar` tworzy pojedynczy plik (*tarball*), który jest konkatenacją plików wskazanych jako argumenty:

Tworzenie tarballa

```
$ tar -cvf tarball.tar somefile1 somefile2 ...
```

Ekstrakcja tarballa

```
$ tar -xvf tarball.tar
```

- Opcje:
  - `-c` tworzenie (*Create*),
  - `-x` ekstrakcja (*eXtract*),
  - `-v` dosadne logi (*Verbose*),
  - `-f` określa nazwę tarballa (*Filename*): `-f tarball.tar` .
- Ponadto `tar`
  - Nie zmienia plików źródłowych
  - Nie dodaje automatycznie magicznych rozszerzeń (`.tar`) przy tworzeniu.

Skompresowane tarballe

- By stworzyć skompresowane archiwum wieloplikowe, można najpierw użyć `tar`, a potem skompresować je przez `gzip`, `bzip2` albo `xz`.

Kompresowanie tarballa za pomocą gzip

```
$ tar -cvf tarball.tar somefile1 somefile2 ...
$ gzip tarball.tar
```

- Alternatywnie, można te kroki skleić używając opcji `-z` (dla `gzip`), `-j` (dla `bzip2`) bądź `-J` (dla `xz`).

Tworzenie skompresowanych tarballi

```
$ tar -czvf tarball.tar.gz somefile1 somefile2 ...
$ tar -cjvf tarball.tar.bz2 somefile1 somefile2 ...
$ tar -cJvf tarball.tar.xz somefile1 somefile2 ...
$ tar -xzvf tarball.tar.gz
$ tar -xjvf tarball.tar.bz2
$ tar -xJvf tarball.tar.xz
```

Skompresowane tarballe z Windows

- Większość narzędzi Windowsa jest w stanie poradzić sobie ze skompresowanymi tarballami.
- Niektóre narzędzia Windowsa mogą działać niepoprawnie w przypadku wielokrotnych rozszerzeń, więc bezpieczniej używać następujących:

  `.tgz` dla tarballi skompresowanych `gzip` ,
  `.tbz` dla tarballi skompresowanych `bzip2` ,
  `.txz` dla tarballi skompresowanych `xz` ,

Plan zajęć

Znajdowanie plików – *find*

Syntax komendy *find*

```
$ find [ścieżka] [warunki] [działania]
```

- Gdzie parametry to:

ścieżka	z perspektywy której poszukiwania będą wykonane. Przeszukiwane są tylko podfoldery wskazanego folderu. Domyślna wartość: bieżący folder,
warunki	które musi spełnić plik by być uznanym za trafienie; można je łączyć za pomocą operatorów logicznych: `!`, `-a` (*and*, domyślne), `-o` (*or*) oraz nawiasów `( )`,
działania	które zostaną przeprowadzone na plikach, które spełniają warunki (domyślnie: wypisuje nazwę pliku)

find – warunki i działania

- Popularne warunki poszukiwania pliku:

  -name, iname po nazwie (bądź nazwie ze zignorowanymi wielkościami liter), z możliwością używania jokerów,

  -user, group po właścicielu i grupie (symbolicznie bądź numerycznie wyspecyfikowanym/-ej),

  -type po typie pliku (`f, d, l, c, b, ...`),

  -perm po uprawnieniach (określonych numerycznie bądź symbolicznie),

  -mtime po czasie ostatniej modyfikacji (`+n` – więcej niż `n` dni, `-n` – mniej niż `n` dni, `n` – dokładnie `n` dni).

- Popularne działania:

  -print wyświetlanie nazw plików, domyślna,

  -delete usuwanie plików,

  -exec *cmd* ; wywołanie komendy `cmd` na plikach, na przykład: `-exec rm {} \;` ma ten sam efekt co `-delete` (backslash pozwala uniknąć interpretowania średnika przez Powłokę).

find – przykłady

Znajdź stosunkowo nowe pliki konfiguracyjne

```
$ find /etc -mtime -30
```

Przesuń stare backupy do katalogu z backupami

```
$ find thisdir -mtime +365 -name "*.bak" -exec mv {}
backdir/ \;
```

Usuń niepotrzenbe pliki interaktywnie

```
$ find thisdir -name ".*~" -exec rm -i {} \;
```