

Technologie Informacyjne i Komunikacyjne R

Linux — przykładowe rozwiązania zadań 1–16

Dla prowadzących (nie rozdawać studentom)

Na podstawie zestawu zadań: Katarzyna Grzelak (TIK-R, 2025)

Opracowanie: Michał Parniak (2026), mparniak@fuw.edu.pl

Rozwiązania są przykładowe; większość zadań da się zrobić na kilka sposobów. Skrypty z zadań 8, 10 i 12–16 zostały przetestowane w `bash` 5.2. Struktury katalogu `~kaste/sum` (zad. 2, 11, 12) nie sprawdzano — polecenia są napisane tak, by nie zależały od niej.

Zajęcia 1

Zad. 1

```
mkdir -p ~/tikr/cwiczenia1
cp /proc/*info* ~/tikr/cwiczenia1/
cd ~/tikr/cwiczenia1
grep 'model name' cpuinfo | head -1      # procesor
grep -c processor cpuinfo                # liczba rdzeni (logicznych)
```

Pliki z `/proc` mają prawa `r--r--r--`, a kopie je dziedziczą — ponowne `cp` kończy się `Permission denied`. Wtedy `cp -f` albo usunięcie kopii (dobry moment, żeby to omówić).

Zad. 2

```
cp -r ~kaste/sum ~/tikr/cwiczenia1/
chmod -R go-rwx ~/tikr/cwiczenia1/sum      # albo: chmod -R u+rwX,go-rwx ...
ls -lR ~/tikr/cwiczenia1/sum
```

Wielkie X nadaje x tylko katalogom (i plikom, które już je mają).

Zad. 3 (w katalogu `~/tikr/cwiczenia1`)

```
tar cvf sum.tar sum
tar czvf sum.tar.gz sum
zip -r sum.zip sum
ls -lh sum.*
tar tzvf sum.tar.gz
mkdir ~/tikr/test && tar xzvf sum.tar.gz -C ~/tikr/test && ls -lR ~/tikr/test
```

Zajęcia 2

Zad. 4

```
quota
cd; du -s * .[!..]* | sort -n | tail -1
cd ~/_work_; du -s * .[!..]* | sort -n | tail -1
```

Zad. 5 Do `~/.bashrc` dopisujemy:

```
alias rm='rm -i'
alias cp='cp -i'
alias mv='mv -i'
alias gedit='gnome-text-editor'
```

Alias zdefiniowany w terminalu działa tylko w nim (i nie jest dziedziczony przez nowe okna). Do `~/.bash_profile`, jeśli nie ma tam odwołania do `.bashrc`: `source ~/.bashrc`. Gdyby student napisał systemowy `.bashrc`: kopia w `/dmj/.snapshot/`, wzorzec w `/etc/skel/.bashrc`.

Zad. 6

```
| cd; du -sh * .[!..]* | sort -h | tail -1 > ~/tikr/cwiczenia1/najwiekszy.txt
```

Zad. 7

```
| ps -ef | grep $USER > procesy.txt      # lepiej: grep "^$USER " (tylko 1. kolumna)
| ps -ef | grep 'ps -ef'                # PPID procesu ps = PID powłoki bash
| echo $$                               # w drugim terminalu
| kill -9 4711                          # z pierwszego terminala, numer z echo $$
```

W wyniku samego `grep $USER` pojawia się też sam proces `grep` i procesy, w których login występuje w linii poleceń (np. ścieżka `/dmj/2026/login`). Rodzicem `bash` jest emulator terminala (`gnome-terminal-server`) lub `sshd`.

Zad. 8 — skrypt szukaj1. Ścieżki zaczynamy od `~`, więc skrypt działa z każdego miejsca.

```
|#!/bin/bash
|mkdir -p ~/tikr/cwiczenia2 2> /dev/null
|cd ~/tikr/cwiczenia2
|find /etc -name '*root*' > pliki1.txt 2> /dev/null
|grep -rl root /etc > pliki2.txt 2> /dev/null
|ls -la ~ > pliki3.txt 2> /dev/null
|cat pliki1.txt pliki2.txt pliki3.txt > pliki.txt 2> /dev/null
|wc -l < pliki.txt > tmp.txt 2> /dev/null
|cat pliki.txt >> tmp.txt 2> /dev/null
|mv tmp.txt pliki.txt 2> /dev/null
```

Zad. 9 (na `tempac`; plik `uzytkownicy_okwf.txt` z `pliki/`, wynik `generuj_uzytkownicy.py`, 1200 linii; przykład dla imienia Jan)

```
|ssh ab123456@tempac.okwf.fuw.edu.pl
|hostname; pwd
|wget -O uzytkownicy.txt <link>
|wc -l uzytkownicy.txt                                # 1200
|cut -d: -f5 uzytkownicy.txt | grep -iw jan | wc -l    # 46 (Anna: 25)
|cut -d: -f5 uzytkownicy.txt | grep -iw jan > nazwiska.txt
|cut -d: -f5 uzytkownicy.txt | cut -d' ' -f1 | grep -ix jan | wc -l  # 35 (Anna: 20)
|cut -d: -f1 uzytkownicy.txt | sort > loginy.txt
```

Typowy błąd: `grep -i jan` bez `-w` daje 56, bo łapie też nazwisko Jankowski. Pliki zapisane na `tempac` łądują w tym samym katalogu domowym (wspólny dla komputerów OKWF).

Zajęcia 3

Zad. 10 — info

```
|#!/bin/bash
|echo "Witaj $USER"
|echo "Pracujesz na komputerze $HOSTNAME"
|echo "Używasz powłoki $SHELL"
|echo "Aktualny katalog to $PWD"
|echo "Liczba plików: $(ls | wc -l)"
```

`$SHELL` to powłoka *logowania*, nie necessarily ta, w której działa skrypt.

Zad. 11

```
|cat sum/*/* > wierszyk.txt                # jeśli zwrotki leżą o jeden poziom niżej
|find sum -type f -exec cat {} + > wierszyk.txt  # niezależnie od głębokości
```

Wersja w dobrej kolejności (zwrotki w plikach `a`, `b`, ...; `*` rozwija się alfabetycznie):

```
#!/bin/bash
mkdir -p zwrotki
for plik in $(find sum -type f); do
    cp "$plik" zwrotki/
done
cat zwrotki/* > wierszyk.txt
```

Zad. 12

```
grep -ri sum sum > sum.txt
```

```
#!/bin/bash
# szukaj2 wzorzec katalog plik_wynikowy
if [ $# -lt 3 ]; then
    echo "Użycie: $0 wzorzec katalog plik_wynikowy"
    exit 1
fi
grep -ri "$1" "$2" > "$3" 2> /dev/null
```

Zad. 13

```
#!/bin/bash
# a)
if [ ! -d zadanie ]; then
    mkdir zadanie
fi
if [ ! -d zadanie/testy ]; then
    mkdir zadanie/testy
fi
cp -f /proc/meminfo zadanie/testy/      # -f: patrz uwaga do zad. 1
```

```
#!/bin/bash
# b) i c): liczba katalogów z argumentu, domyślnie 5
n=${1:-5}
sciezka=.
for i in $(seq 1 $n); do
    sciezka=$sciezka/test$i
    if [ -d "$sciezka" ]; then
        echo "Katalog $sciezka już istnieje"
    else
        mkdir "$sciezka"
    fi
done
```

Zad. 14 — kopiuj

```
#!/bin/bash
if [ -z "$1" ]; then
    echo "Użycie: $0 katalog_docelowy katalog_zrodlowy rozszerzenie"
    exit 1
fi
if [ -d "$1" ]; then
    echo "Katalog $1 już istnieje"
else
    mkdir "$1"
fi
cp "$2"/*."$3" "$1/"
```

Bez cudzysłowów wokół \$1 nazwa moja kopia rozpada się na dwa argumenty.

Zad. 15

```
#!/bin/bash
plik=~/.aliasy.sh
if [ -e $plik ]; then
    echo "Plik ~/.aliasy.sh istnieje"
    cat $plik
else
    echo "Plik ~/.aliasy.sh nie istnieje -- tworzę go"
    cat << 'KONIEC' > $plik
alias rm='rm -i'
alias cp='cp -i'
alias mv='mv -i'
KONIEC
fi
```

Wczytanie aliasów do bieżącego terminala: `source ~/.aliasy.sh` (uruchomienie `./aliasy.sh` nic nie da — aliasy zostałyby w powłoce potomnej).

Zad. 16 — kalkulator

```
#!/bin/bash
read -p "Pierwsza liczba: " a
read -p "Druga liczba: " b
read -p "Działanie (+ - * /): " op
case "$op" in
    "+") echo "$a + $b = $((a + b))" ;;
    "-") echo "$a - $b = $((a - b))" ;;
    "*") echo "$a * $b = $((a * b))" ;;
    "/" ) if [ "$b" -eq 0 ]; then
            echo "Nie dzielimy przez zero!"
            exit 1
        fi
        echo "$a / $b = $((a / b))" ;;
    *)    echo "Nieznane działanie: $op"
        exit 1 ;;
esac
```

Bez cudzysłówów `*` w `case` pasuje do wszystkiego, a w `echo $op` rozwija się w listę plików. Dzielenie przez zero bez zabezpieczenia: `bash` zgłasza błąd `division by 0`. Dzielenie jest całkowite ($7 / 2 = 3$).