Systemy operacyjne

Mariusz Żynel
mariusz@math.uwb.edu.pl
http://math.uwb.edu.pl/~mariusz/

Uniwersytet w Białymstoku

2024/2025

Co potrafi Unix?

Powłoka shell

- Interfejs użytkownika w postaci interpretera poleceń
- Interaktywna sesja między użytkownikiem a systemem operacyjnym
- Automatycznie uruchamiana w momencie logowania do systemu
- Ukrywa techniczne detale jądra systemu operacyjnego
- Działa w przestrzeni użytkownika, nie w jądrze
- Polecenia wprowadza się bezpośrednio na terminalu lub z pliku
- Implementacje:
  - Thompson shell (`sh`) – pierwsza wersja shella z 1971 roku
  - Bourne shell (`sh`) – unowocześniona w 1979 wersja shella Thompsona
  - C shell (`csh`) – shell ze składnią poleceń wzorowaną na C
  - Korn shell (`ksh`) – rozszerzenie `sh`, zapożyczenia z `csh`
  - Bourne-Again shell (`bash`) – najbardziej popularny, napisany dla GNU

Co czyni shell tak potężnym?

- Metaznaki (wildcarding, globbing)
- Przekierowania/redyrekcje wejścia/wyjścia (`I/O` redirections)
- Potoki (pipelines)
- Zmienne
- Here documents
- Command substitution
- Instrukcje warunkowe (`if then else`, `switch`)
- Instrukcje iteracyjne, pętle (`for`, `while`, `until`)

Podstawowe polecenia i programy narzędziowe

`ls` (list) wyświetla zawartość katalogu

`pwd` (path of working directory) wyświetla ścieżkę katalogu roboczego

`cd` (change directory) przechodzi do innego katalogu

`cp` (copy) kopiuje pliki

`mv` (move) przenosi pliki

`mkdir` (make directory) tworzy katalogi

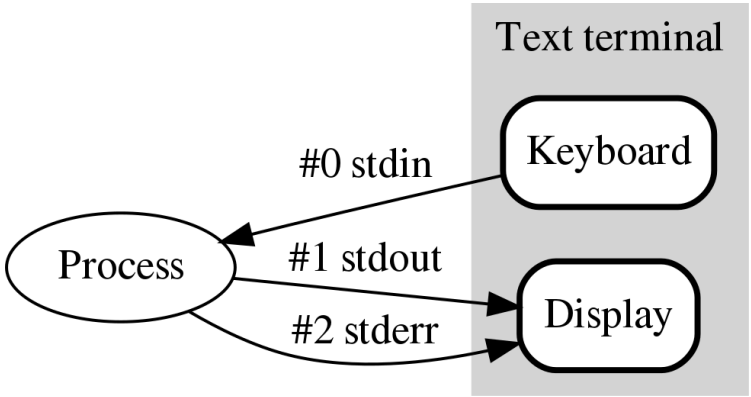
`rm` (remove) usuwa pliki, które nie są katalogami

`rmdir` (remove directory) usuwa katalogi jeśli nie są puste

`cat` (concatenation) łączy i wyświetla podane pliki

Standardowe strumienie

- Z każdym procesem związane są trzy specjalne kanały `I/O`
- Dowiązanie następuje w momencie uruchomienia programu
- Standardowe strumienie `I/O` służą do komunikacji z otoczeniem



`stdin` (deskryptor 0) – standardowy strumień wejścia, dane wejściowe, zwykle tekst, przekazywane do programu

`stdout` (deskryptor 1) – standardowy strumień wyjścia, strumień, do którego program zapisuje dane wynikowe

`stderr` (deskryptor 2) – standardowy strumień błędów, wykorzystywany do wyświetlania komunikatów o błędach i informacji diagnostycznych

Shell – metaznaki

Przy pomocy metaznaków (wildcard characters) możemy tworzyć wzorce (glob patterns) nazw plików. Shell `sh` rozumie następujące metaznaki:

- `*` zastępuje dowolny ciąg znaków, także pusty
- `?` zastępuje jeden dowolny znak
- `[abc]` zastępuje jeden z wyszczególnionych znaków
- `[!abc]` zastępuje jeden znak spoza wyszczególnionych znaków
- `[a-z]` zastępuje jeden znak z podanego zakresu
- `[!a-z]` zastępuje jeden znak spoza podanego zakresu

Uwaga

Znak / oraz . jeśli występuje na początku nazwy pliku nigdy nie zostaną zastąpione.

Shell – przekierowania/redyrekcje wejścia/wyjścia

`<plik` traktuje `plik` jako strumień wejściowy

```
sh < skrypt.sh
```

`>plik` traktuje `plik` jako strumień wyjściowy

```
ls -l > /tmp/lista_plikow
date > /var/tmp/logfile
```

`>>plik` traktuje `plik` jako strumień wyjściowy dopisując do niego

```
pwd >> /var/tmp/logfile
echo "nie cierpie Unixa" >> moje_motto
```

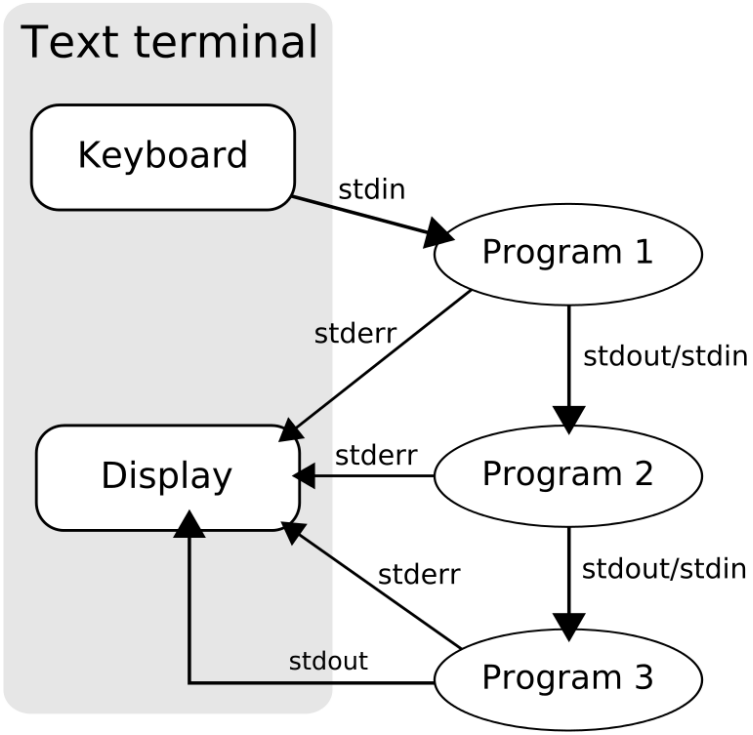
`2>&1` przekierowanie strumienia błędów na standardowe wyjście

`>plik 2>&1` przekierowanie strumienia wyjściowego i błędów na `plik`

`>>plik 2>&1` dopisywanie strumienia wyjściowego i błędów do `plik`

Shell – potoki

- Potok to ciąg poleceń rozdzielonych znakiem | (pipe)
- Standardowe wyjście każdego z poleceń poza ostatnim jest skojarzone ze standardowym wejściem następnego polecenia
- Każde z poleceń uruchamiane jest jako osobny proces
- Potok kończy się, gdy zakończone zostanie ostatnie polecenie
- Kod wyjścia potoku jest taki jak kod ostatniego polecenia




```
cat *.php | grep -v ^$ | wc -l

tar cf - -C /data/http . | tar xf - -C /backup/http

fgrep '03/Mar/2020' access.log | awk '{print $1}' | \
   sort | uniq -c | sort -n -r | head -15
```

Shell – here documents

- Here document to literał strumieniowy lub wielolinijkowy literał tekstowy
- Here document rozpoczyna się od znaków << i następującego po nich identyfikatora ograniczającego, który jest dowolnym napisem
- Od nowego wiersza wprowadzany jest here document
- Here document kończy się wprowadzeniem tego samego identyfikatora ograniczającego w nowym wierszu
- Składnia nie jest przypadkowa bo zawartość here document trafia na standardowe wejście polecenia poprzedzającego

```
tr [a-z] [A-Z] <<EOF
Nie cierpie Unixa
Nie znosze komputerow
Przydalaby sie przerwa
EOF
```

Shell – zmienne

- Zmienna to wydzielony fragment pamięci, do którego odwołujemy się poprzez nazwę symboliczną, czyli identyfikator zmiennej
- Zawartość tego fragmentu pamięci to wartość zmiennej
- Zmienne w shellu nie mają typów i nie są deklarowane przed użyciem
- Dozwolone znaki w identyfikatorach zmiennych to: `a-zA-Z0-9_`
- Wielkość liter w identyfikatorach ma znaczenie (case sensitive)
- Aby odczytać wartość zmiennej identyfikator poprzedzamy znakiem `$`

```
HW="Hello world!"
echo $HW

X=1+1
echo $X
```

Shell – command substitution

- Command substitution to mechanizm pozwalający wykonać polecenie i wstawić jego wynik w linii poleceń jako argument innego polecenia
- Command substitution wykonujemy wstawiając całe polecenie wraz z jego argumentami w lewe apostrofy ‘...‘

```
NBLINES=‘cat *.php | grep -v ^$ | wc -l‘

i=‘expr $i + 1‘

cp ‘grep -l malloc *.c‘ /tmp/malloc\_files
```

Shell – instrukcje warunkowe

```
if [ -f $1 ]
then
    echo "Plik $1 istnieje"
else
    echo "Plik $1 nie istnieje"
fi
```

```
case $1 in
    -d) echo "Debug mode on"
      ;;
    -v) echo "Verbose mode on"
      ;;
     *) echo "usage: $0 [-dv]"
      ;;
esac
```

Shell – instrukcje iteracyjne, pętle

- Pętla `for`

```
for f in ‘grep -l malloc *.c‘
do
  sed s/malloc/valloc/ $f > /tmp/x
  cp /tmp/x $f
done
```

- Pętla `while`

```
grep -l malloc *.c | while read f
do
  sed s/malloc/valloc/ $f > /tmp/x
  cp /tmp/x $f
done
```

- Pętla `until`

```
sum=0
until [ "$x" = "0" ]
do
  read x
  sum=‘expr $sum + $x‘
done
echo "suma: $sum"
```