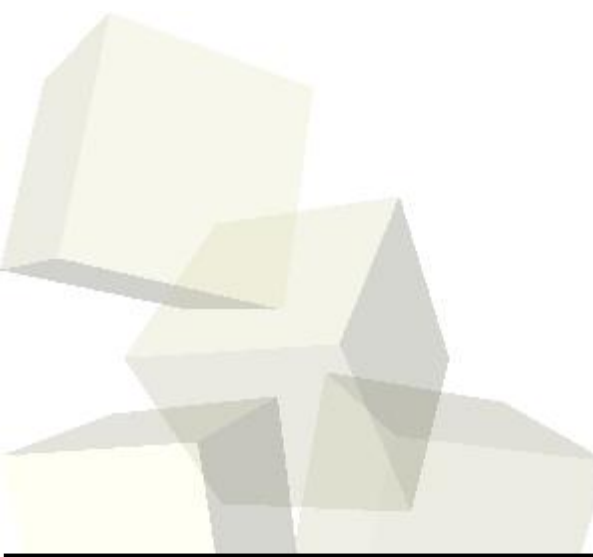




# Úvod do Operačních Systémů

## 10. cvičení

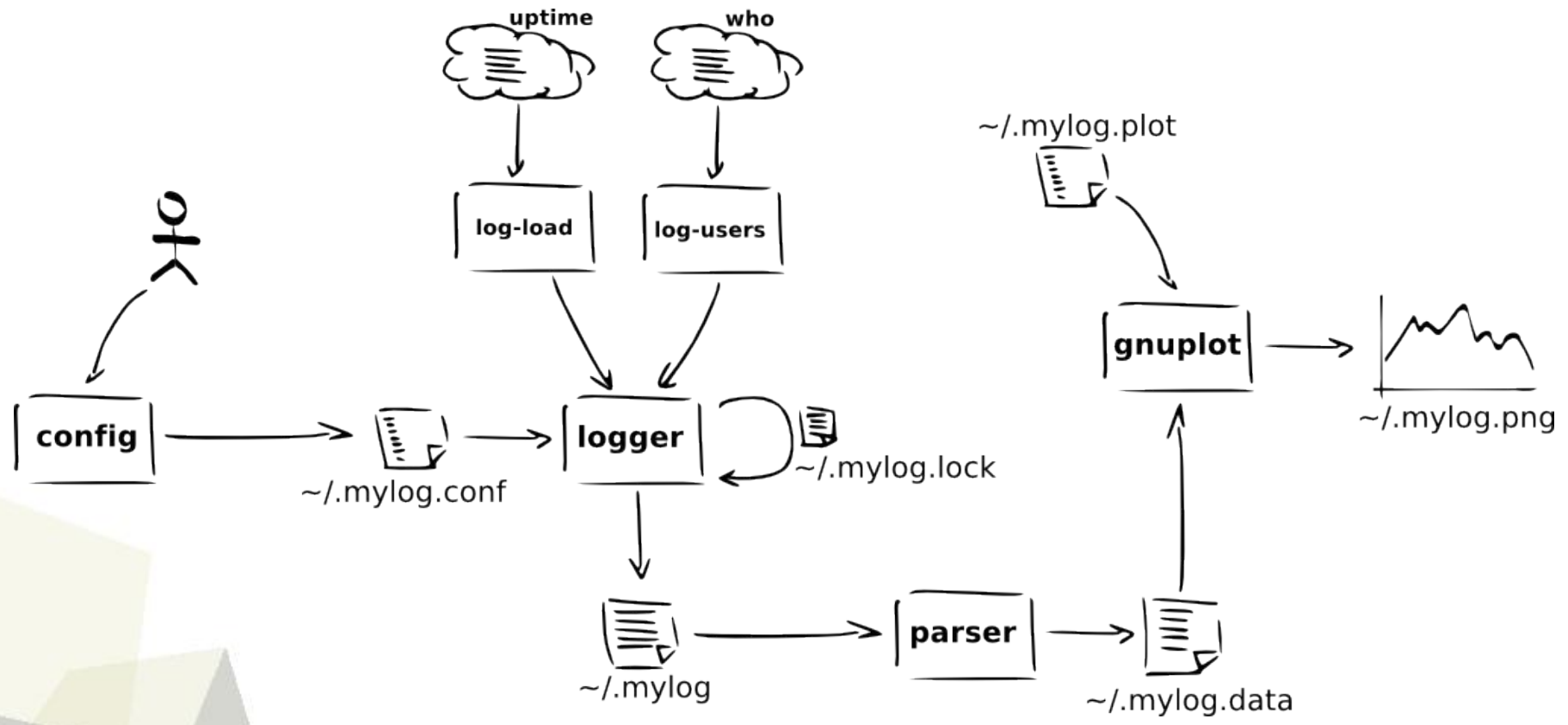
Uživatelský vstup, zpracování přepínačů,  
psaní a ladění skriptů, plánování úloh.





## Skript pro logování informací o systému, uložení konfigurace, transformaci logu.

- Psaní a ladění skriptů
  - #!
  - set
- Zpracování přepínačů
  - getopts
- Uživatelský vstup
  - read
- Plánování úloh
  - crontab





Skript bude logovat informace o zatížení systému (load) (průměr za 1, 5 a 15 min) a počtu přihlášených uživatelů (celkem a unikátních).

- Formát log souboru

```
[27/11/2006-07:30:00] Load: 0.75 0.36 0.12
```

```
[27/11/2006-07:30:00] Users: 38 29
```

- Informace o zatížení systému

```
uptime ...
```

- Počet přihlášených uživatelů a počet unikátních uživatelů

```
who ...
```



- Interpret skriptu (první řádek skriptu)

```
#!/bin/bash
```

- Výpis prováděných řádků skriptu

```
#!/bin/bash -v
```

```
set -v
```

- Výpis prováděných řádků skriptu po expanzi

```
#!/bin/bash -x
```

```
set -x
```

- Nastavení přístupových práv a spuštění skriptu

```
chmod +rx logger
```

```
./logger
```



# Transformační funkce

- Volání logovacích skriptů, transformace výstupu pomocí funkce log, kde vstupem je jméno informace a samotná informace.

```
filename=.mylog
```

```
function log() {
```

```
...
```

```
}
```

```
# Logovani urcenych informaci
```

```
log Load hodnoty...
```

```
log Users hodnoty...
```



- Skript při startu zkontroluje, zda neexistuje zvolený soubor (zámek). Pokud ne, vytvoří jej a vloží do něj PID skriptu.

```
LOCK_FILE=$HOME/.mylog.lock
```

```
if locked
  then
    exit 2
  else
    echo $$ > "$LOCK_FILE"
    trap "rm $LOCK_FILE; exit" 2 3 15
```

```
fi
```

```
...
```

```
rm "$LOCK_FILE"
```



- Pokud soubor existuje, ale obsahuje PID procesu, který nepatří tomuto skriptu, soubor se ignoruje.

```
function locked() {  
    if [ -f "$LOCK_FILE" ] \  
        && ps -u $USER -o pid,args \  
            | grep "^ *`cat $LOCK_FILE` .* logger" \  
                >/dev/null  
  
    then  
        echo "Lock file found! (PID=`cat $LOCK_FILE`)"  
        return 0  
    else  
        return 1  
    fi  
}
```





# Volby z příkazové řádky I

```
VERBOSE=0
```

```
CONFIG=$HOME/.mylog.conf
```

```
LOCK_FILE=$HOME/.mylog.lock
```

```
USAGE="Usage: $0 [-h] [-vd] [-c config_file]"
```

```
HELP="
```

```
    -h      this help
    -v      verbose mode (use multiply for more verbosity)
    -d      daemon mode (check lock file)
    -c conf use alternative configuration file
```

```
"
```



# Volby z příkazové řádky II +

```
while getopt hvc: opt
do
  case $opt in
    h) #display help
       echo "$USAGE"; echo "$HELP"; exit 0;;
    c) #config file
       if [ -f "$OPTARG" -a -w "$OPTARG" ]
         then CONFIG=$OPTARG
          else echo "$OPTARG: wrong config file" >&2; exit 2
         fi;;
    v) #verbose mode
       ((VERBOSE++))
       ((VERBOSE>0)) && set -v
       ((VERBOSE>1)) && set -x
       ;;
    \?) #err - unknown option
        echo "$USAGE" >&2; exit 2
        ;;
  esac
done ; shift `expr $OPTIND - 1`
```



- Konfigurační soubor obsahující nastavení ve formátu *název\_proměnné=hodnota*

```
#MyLog config file
#Modified: 27.lis 2006 (07:30:00)
#Modified by barinkl

#filename of log file
#filename=/home/k336/barinkl/.mylog
filename=/home/k336/barinkl/.mylog

#log system load
#load=y
load=y

#log number of users
#users=y
users=y
```



# Konfigurační soubor, vytvoření I

- Skript vytvářející konfigurační soubor podle vstupu uživatele. Vstup je čten pomocí příkazu `read` .

```
#cesta k log souboru a jeho jmeno
until [ -f "$filename" -a -w "$filename" ]
do
    echo -n "Log filename [$HOME/.mylog]: "
    read filename junk
    echo "${filename:= $HOME/.mylog}"
    if [ "`echo $filename | cut -c1`" != / ]
    then
        filename="$PWD/$filename"
    fi
    [ -f "$filename" ] \
    || { touch "$filename"; chmod u+w "$filename"; }
done
```



# Konfigurační soubor, vytvoření II

- Dotaz na logování zátěže (load) – možnosti y/n

```
while ! echo "$load" | grep '^[yYnN]$'
do
    echo -n "Log load [y]/n? "
    read load junk
    [[ -z "$load" ]] && load=y
done
```

- Dotaz na logování počtu uživatelů (users) – možnosti y/n

```
while ! echo "$users" | grep '^[yYnN]$'
do
    echo -n "Log users [y]/n? "
    read users junk
    [[ -z "$users" ]] && users=y;
done
```



# Konfigurační soubor, vytvoření III

```
cat <<KONEC >"$CONFIG"  
#MyLog config file  
#Modified: `date '+%d.%b %Y (%T)'\`  
#Modified by $USER  
  
#filename of log file  
#filename=$HOME/.mylog  
filename=$filename  
  
#log system load  
#load=y  
load=$load  
  
#log number of users  
#users=y  
users=$users  
KONEC
```



# Načtení konfigurace I

- Načtení funkce ze souboru readconf do skriptu
  - . ./readconf
- Načtení konfigurace ze souboru. Funkce je v samostatném souboru readconf.

```
function readconf() {
    err=0
    while [ $# -gt 0 ]
    do

        case "$1" in
            filename)
                filename=`sed -n '/^filename=/s/[^=]*=//p' $CONFIG \
                    | tail -1`

                if ! [ -f "$filename" -a -w "$filename" ]
                then err=1;
                fi

                ;;
        esac
    done
}
```



# Načtení konfigurace II

```
load)
load=`sed -n '/^load=/s/[^=]*=//p' $CONFIG \
| tail -1`
```

```
if [[ "$load" != [yYnN] ]]; then err=1; fi
;;
```

```
users)
users=`sed -n '/^users=/s/[^=]*=//p' $CONFIG \
| tail -1`
```

```
if [[ "$users" != [yYnN] ]]; then err=1; fi
;;
```

```
*) return 2;;
```

```
esac
```

```
shift
```

```
done
```

```
return $err
```

```
}
```

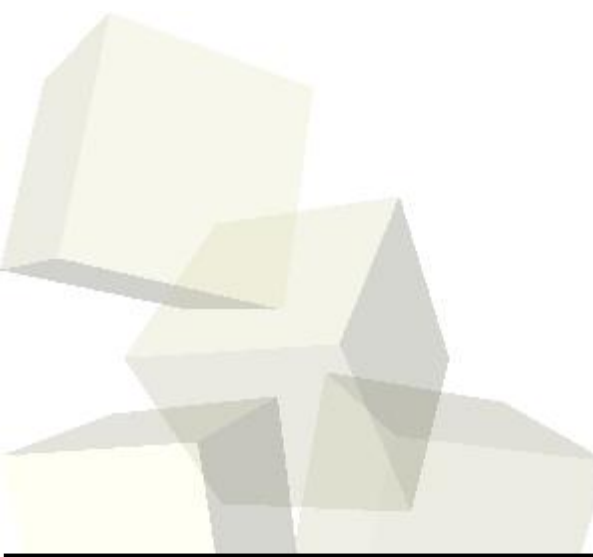




# Použití načtení konfigurace

- Načtení proměnných z konfiguračního souboru

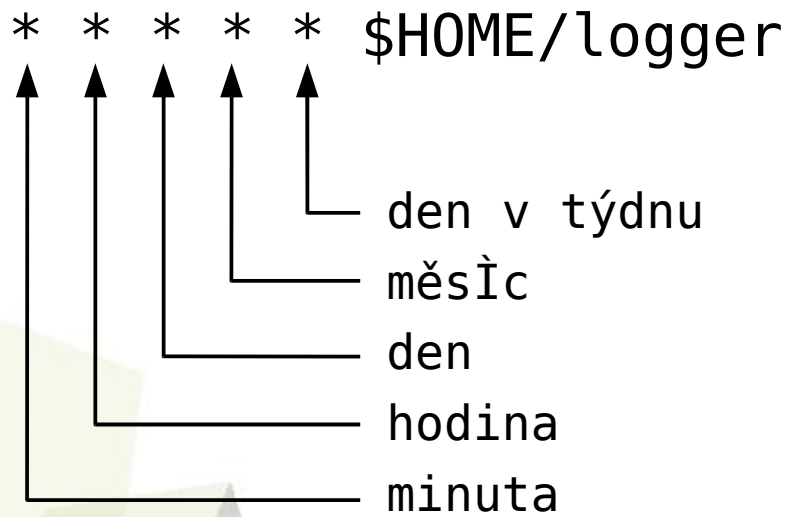
```
# Načtení konfigurace
if ! readconf filename load users
then
    echo "Error reading config file"
    exit 2
fi
```





- Nastavení a vypsání opakovaného spouštění úloh

```
export EDITOR=vi
crontab -e
```



- Možné hodnoty:

*	<i>každá hodnota</i>
*/2	<i>každá 2. hodnota</i>
5	<i>hodnota 5</i>
2,7	<i>hodnoty 2 a 7</i>
1-5	<i>hodnoty 1,2,3,4,5</i>

```
crontab -l
```



# Vykreslení grafu z dat v logu +

- Transformace dat z logu

```
awk '/Load:/ {L1=$3; L2=$4; L3=$5}
     /Users:/ { print $1,L1,L2,L3,$3,$4}
     ~/.mylog > .mylog.data
```

- Vykreslení grafu pomocí příkazu gnuplot

```
gnuplot .mylog.plot
```

- Zobrazení grafu v obrázku

```
display .mylog.png
```



# Skript pro gnuplot ++

```
set terminal png small color
set size 1.5
set output ".mylog.png"
set rmargin 3
set lmargin 6
set tmargin 1
set bmargin 1.5
set nograd
set multiplot
set xdata time
set timefmt "[%d/%m/%Y-%H:%M:%S]"

set format x ""
set size ratio 0.5
plot [][0:] \
    '.mylog.data' using 1:2 title '1 min avg' with lines, \
    '' using 1:3 smooth bezier title '5 min avg' with lines, \
    '' using 1:4 smooth bezier title '15 min avg' with lines

set format x "%H:%M"
set origin 0,0.99
set size ratio 0.25
plot [][0:] \
    '.mylog.data' using 1:5 title 'users' with lines, \
    '' using 1:6 title 'uniq users' with lines
```



# Příprava na příští cvičení

- Upravte skripty z tohoto cvičení tak aby
  - byly uživateli hlášeny chyby při práci se soubory (neexistence, nedostatečná práva, špatný obsah, ...),
  - skripty i soubory mohly být uloženy v libovolném adresáři (odstranění absolutních cest),
  - skript přijímal argumenty z příkazové řádky, které by uváděly názvy logovacích funkcí (skriptů). Argumenty by pak měly přednost před nastavením z konfiguračního souboru.
  - jméno log souboru mohlo být předáno jako argument přepínače -o z příkazové řádky,
  - přibyla informace, která se bude také logovat.
- Vytvořte skript a naplánujte jeho pravidelné spouštění, který by generoval do určeného adresáře grafy z log souboru za zvolené období.