F4270 UNIX, počítačové sítě (jaro 2016), Cvičení 8

Příkazy: `expr, trap, ps, ps -o, kill, sleep, date, eval, SIGKILL, SIGINT, SIGTERM, SIGQUIT`.

A1 Napište skript, který dostane tři parametry, všechny číselné, a vypíše na standardní výstup čísla oddělená mezerami počínaje `$1`, konče `$2` a s krokem `$3`. Pokud skript dostane jen dva parametry, pak se implicitně použije krok 1.

```
#!/bin/sh

if [ $# -lt 2 ] || [ $# -gt 3 ]
then
   echo Špatný počet parametrů
   exit 1
fi

krok=${3:-1}

x=$1
while [ $x -le $2 ]
do
   echo $x
   x=`expr $x + $krok`
done
```

A2 Totéž jako předchozí příklad, ale čísla budou každé na zvláštní řádce a budou zarovnaná na počet míst daný délkou nejdelšího čísla (tj. délkou `$2`). Čísla budou zarovnaná nulami na začátku. Tj. například `skript 0 10 2` by vypsal:

```
[guy@nix ~]$ skript 0 10 2
00
02
04
06
08
10
```

```
#!/bin/sh
if [ $# -lt 2 ] || [ $# -gt 3 ]
then
   echo Špatný počet parametrů
   exit 1
fi

krok=${3:-1}

x=$1
delka=${#2}
while [ $x -le $2 ]
do
   printf "%0${delka}d\n" $x
   x=`expr $x + $krok`
done
```

A3 Napište skript, který přejmenuje všechny soubory v aktuálním adresáři s příponou jpg na soubory s touž příponou ale s čísly místo názvů, kde čísla budou zarovnaná na tolik míst, kolik je třeba pomocí nul. (Využijte toho, co jste už udělali v jednom z předchozích příkladů.) Tj. například pokud jsou v adresáři soubory `alice.jpg`, `bob.jpg` `cyril.jpg` a `daniel.jpg`, přejmenoval by je popořadě skript na `1.jpg`, `2.jpg`, `3.jpg` a `4.jpg`. Pokud by těchto souborů bylo alespoň deset ale méně než 100, byly by pojmenovány jako `01.jpg`, `02.jpg`, `03.jpg`, `04.jpg`, atd.

```
#!/bin/sh
x=1
pocet=`ls ./*.jpg | wc -l`

# Tohle je pro případ, že by tam byl nějaký soubor s příponou *.jpg a
# číselným názvem, mohlo by to selhat:
for soubor in *.jpg
do
   mv "$soubor" "x$soubor"
done

for soubor in *.jpg
do
   mv "$soubor" `printf "%0${#pocet}d\n" $x`.jpg
   x=`expr $x + 1`
done
```

A4 Napište skript, který očekává na standardním vstupu řádky ve tvaru: `a+b=c` kde `a`, `b` a `c` jsou čísla, skript načte pro každou řádku tato tři čísla a zkontroluje, jestli je rovnost správná. Pokud není správná, ohlásí chybu s číslem řádky, na které k ní došlo. Na závěr vypíše celkový počet chyb.

```
#!/bin/sh
IFS="+="
radka=0
chyba=0
while read a b c
do
   radka=$(expr $radka + 1)
   if [ "$(expr "$a" + "$b")" -ne "$c" ]
   then
      printf "Chyba na řádce %d (%d+%d<>%d).\n" $radka $a $b $c
      chyba=$(expr $chyba + 1)
   fi
done
printf "Počet zkontrolovaných řádek: %d\n" $radka
printf "Počet chybných řádek: %d\n" $chyba
```

A5 Napište skript, který na vstupu očekává řádky tvaru:
`cil : z1 z2 .... zn`
kde `cil`, `z1`, `z2`, ..., `zn` jsou názvy souborů. Ze vstupu vypíše skript ty cíle, které jsou starší než některý ze souborů `z1`, ..., `zn` na témže řádku. Předpokládejte, že v názvech souborů nejsou mezery. Můžete použít soubory z úlohy A3 a pomocí `touch -t` nastavit informace o modifikaci vybraných souborů.

```
#!/bin/sh
```

```
#!/bin/sh
while read cil dvojtecka soubory
do
  echo "#########"
  set $soubory
  echo cil: $cil
  echo soubory: $@
  newer=`find "$@" -newer "$cil" -prune`
  if [ $newer ]
  then
   echo $newer : newer then $cil
  fi
done
```

A6 Napište skript, který bude očekávat vstup na standardním vstupu, načtený vstup pouze zopakuje na svůj standardní výstup. Pokud skript dostane signál `SIGKILL`(9), tak samozřejmě skončí, ale pokud dostane signál `SIGTERM` (15), tak vypíše pouze číslo toho signálu, který takto obdržel a pokračuje dál. Pro `SIGQUIT` (3) vypise navic posledni vstup;

`SIGINT` (2) (to je ten, který je poslán pomocí `Ctrl-c`), nechť je ignorován.

```
#!/bin/sh

function clean_up {
  echo 'signal:' $1
}

echo "#My pid" $$

trap "clean_up 15" TERM
trap 'echo "signal:" 3; echo "Last line:" $x' QUIT
trap '' 2 # ignoruj SIGINT

while true
do
 read x
  echo $x
done
```

A7 Napište skript, který dostane signál a jméno programu, poté pošle signál všem procesům daného programu. Můžete předpokládat, že se program jmenuje normálně a název neobsahuje žádné divné znaky (mezeru ale obsahovat může).

```
#!/bin/sh

[ $# -eq 2 ] || {
   echo "Očekávám dva parametry, číslo signálu a název programu";
   exit 1
}

signal="$1"
program="$2"

ps -e -o pid= -o comm= |\
sed -n 's/^ *\([[:digit:]]\{1,\}\) \{1,\}'"$program"'$/\1/p' |\
xargs kill -$signal

# Případně totéž pomocí while cyklu:
ps -e -o pid= -o comm= | while read pid cmd
do
   if [ "$cmd" == "$2" ]
   then
      kill -$signal $pid
   fi
done
```

A8 Napište skript, který dostane jeden číselný parametr s počtem sekund. Skript potom počká zadaný počet sekund a skončí. Pokud skript obdrží signál `SIGINT` (2: `Ctrl-C`), tak vypíše, kolik sekund už od začátku uběhlo. Pokud obdrží signál `SIGQUIT` (3 - `Ctrl-\`), tak vypíše, kolik ještě zbývá sekund do konce, ale svůj běh nezastaví. Pokud skript obdrží signál `SIGTERM` (15), vypíše, kolik uběhlo sekund od začátku, kolik zbývá do konce a skončí. Návratová hodnota skriptu je buď `0`, pokud skončil ve správný čas, nebo zbývající počet sekund, pokud skončil kvůli `SIGTERM`.

```
#!/bin/sh

die() {
   echo "$1"
```

```
   exit 1;
}
# Vystačíme si s hodinami, minutami a sekundami,
# předpokládáme tedy, že interval bude kratší než den a že v čase
# čekání nebude půlnoc.
now() {
   vzorec=$(date +"%H \* 3600 + %M \* 60 + %S")
   eval expr $vzorec
}

[ $# -eq 1 ] || die "Špatný počet parametrů"
expr "$1" : '[[:digit:]]*$' >/dev/null || die "Parametr není číslo"

start=$(now)
konec=$(expr $start + $1)

trap 'expr $(now) - $start' 2
trap 'expr $konec - $(now)' 3
trap 'expr $(now) - $start; kill $! 2>/dev/null; exit' 15

echo 'my pid:' $$

while [ $(now) -lt $konec ]
do
   sleep $(expr $konec - $(now) ) &
   wait $!
   [ $? -gt 128 ] && kill $! 2>/dev/null
done
```