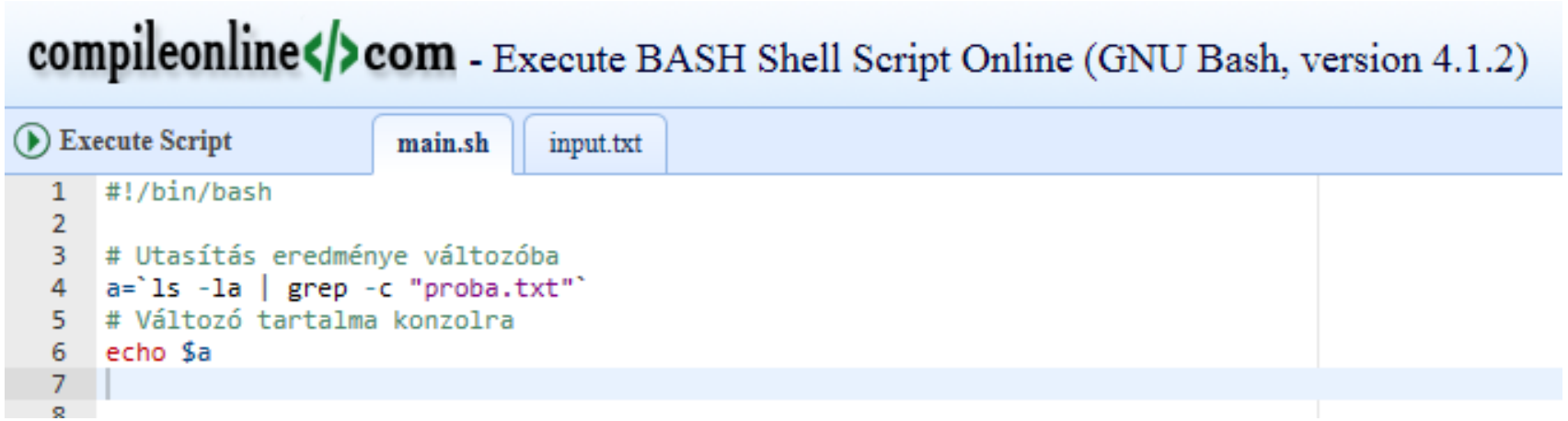
Szkriptek készítése

Szkriptek készítése

- Szkript – shell utasításokat tartalmazó program
- #!/bin/bash
- … (BASH utasítások)
- chmod ugo+x szkript.sh
- Indítás: ./szkript_név.sh
- Indítás: /bin/bash szkript_név.sh

Online bash szkript értelmező



https://www.tutorialspoint.com/execute_bash_online.php

Változók$_1$

- Utasítás eredménye változóba
  - u=`id -u`
  - a=`ls -la | grep -c "proba.txt"`
  - b=`ls -la | grep -c "txt"`
- Változóban tárolt adat átadása parancsnak bemenő paraméterként
  - echo $Vált | parancs
- Változó tartalma konzolra
  - echo $u
- Kiírás a konzolra úgy, hogy a végén ne legyen újsor jel. Pl. adatbekérésnél lehet előnyös
  - echo -n "Kérem a számot:"
  - read a

Változók$_2$

- Numerikus változó deklarálása
  - declare -i szam
- Konstans deklarálása
  - declare –r v=3
- Tömb deklarálása
  - Tomb_nev[index_ertek]=value (indirekt)
  - declare -a Tomb_nev (explicit)
  - Tomb_nev=(value1 value2 ... valueN) (értékadással)

Feltételes elágazás

```
#!/bin/bash
U=`id -u`
echo $U
if [ $U -ne 0 ]
then
  echo "A szkript futtatásához rendszergazdai jogosultság kell!" >&2
  exit 1
fi
```

Feltételes elágazás

```
if [ $a -gt 0 ]
then
  echo "A proba.txt megtalálható az aktuális könyvtárban"
elif [ $b -gt 0 ]
then
  echo "A proba.txt nem található meg az aktuális könyvtárban, de vannak más txt nevűek"
else
  echo "A proba.txt nem található meg az aktuális könyvtárban"
fi
```

Relációs opertátorok

-eq - Equal to

-lt - Less than

-gt - Greater than

-ge - Greater than or Equal to

-le - Less than or Equal to

Negálás: !

Állománnyal kapcsolatos vizsgálatok

-f file – igaz, ha létezik és közönséges állomány

-r file – igaz, ha létezik és olvasható

-w file – igaz, ha létezik és írható

-x file – igaz, ha létezik és végrehajtható

-d file – igaz, ha létezik és könyvtár

-s file – igaz, ha létezik és mérete nagyobb mint 0 bájt

Sztring vizsgálat

-n str - igaz, ha str nem null sztring

-z str - igaz, ha str null sztring

str1 == str2 - igaz, ha azonosak

str - igaz, ha str tartalmaz adatot és nem null

str1 != str2 - igaz, ha különbözőek

Ha a feltételvizsgálatnál egy változó tartalmára sztringként kívánunk hivatkozni, akkor " "-idézőjelek közé kell tenni a hivatkozást: "$a"

Feltételek összekapcsolása

-a - és kapcsolat

-o - vagy kapcsolat

Többszörös elágazás

Szétválasztja a szóközöknél és a negyedik elemet veszi, majd szétválasztja a vesszőnél és az első elemet veszi

```
a=`date |cut -d" " -f4 |cut -d"," -f1`
```

CASE szerkezet

```
case $a in
vasárnap) c=7;;
hétfő) c=1;;
kedd) c=2;;
*) c=0;;
esac
echo $c
echo "Ma a hét $c. napja van"
```

Paraméterek

- Kiíratjuk az átvett paramétereket
- echo $1 $2 $3
- A szkript neve
- echo $0
- A paraméterek száma
- echo $#
- All of the positional parameters, seen as a single word
- echo $*

Kifejezések

- $[kifejezés]
- $((kifejezés #))
- echo $a+$b = $(($a + $b))

- Változó behelyettesítés

Paraméterek

```
#!/bin/bash
Kapcsolo="a|b|c"
if [ $# -eq 0 ]
then
  echo "Hiányzó paraméterek"
  echo "Helyes használat: "
  echo "`basename $0` --help"
  echo "`basename $0` -$Kapcsolo"
  exit 1
elif (( $# > 1 ))
  then
    echo "Csak egy paraméter adható meg!"
    exit 2
fi

SugoSzoveg="..."
case $1 in
"--help") echo $SugoSzoveg;;
"-a")  echo "a";;
"-b")  echo "b";;
"-c")  echo "c";;
*) echo "Hibás paraméter!";;
esac
```

Paraméterek

```
hallgato@ubuntu-server:~$ ./parameterek.sh
Hiányzó paraméterek
Helyes használat:
parameterek.sh --help
parameterek.sh -alblc
hallgato@ubuntu-server:~$ ./parameterek.sh b
Hibás paraméter!
hallgato@ubuntu-server:~$ ./parameterek.sh -b
b
```

Ciklus - WHILE

WHILE ciklus

Várakozunk, amíg okoska ki nem lép

```
while [ `w | grep -c okoska` -gt 0 ]
do
sleep 5
done

echo "Okoska  kilépett!"
```

Ciklus - UNTIL

UNTIL ciklus

Várakozunk, amíg valaki el nem indítja a Midnight Commander-t (pontosabban egy olyan programot, aminek a neve mc-vel kezdődik)

```
until [ `ps -e | grep -c " mc"` -gt 0 ]
do
  sleep 5
done

echo "Valaki elindította a Midnight Commander-t!"
```

Ciklus - FOR

```
# FOR ciklus
# Numerikus ciklusváltozóval
for i in 1 2 3 4 5 6 7
do
  echo "x értéke $i";
done

# A keresési útvonalakból egy tömböt csinál, aminek minden eleme egy elérési út
# A :-okat újsor jelekre cseréli
Tomb=$(echo $PATH | tr ":" "\n")
# Sorra veszi a tömb elemeit és kiírja a képernyőre
for var in $Tomb
do
  echo $var
done
```

Fájlnevek és tulajdonosok

```
#!/bin/bash
for f in *
do
 ls -l "$f" | awk '{ print "Fájl: " $9 " Tulajdonos: " $3 }'
done
```

```
hallgato@ubuntu-server:~$ ~/fajlok
Fájl: apparmor_status Tulajdonos: root
Fájl: fajlok Tulajdonos: hallgato
Fájl: netstat_eredm Tulajdonos: root
Fájl: nmap_eredmeny Tulajdonos: root
Fájl: parameterek.sh Tulajdonos: hallgato
Fájl: roote Tulajdonos: hallgato
hallgato@ubuntu-server:~$ _
```

For lista nélkül

```
#!/bin/bash
# forlistanelkul.sh

echo "A paraméterek száma: $#"
echo -n "Az egyes paraméterek: "
for i
do
 echo -n "$i, "
done
echo
```

```
hallgato@ubuntu-server:~$ ./forlistanelkul.sh
A paraméterek száma: 0
Az egyes paraméterek:
hallgato@ubuntu-server:~$ ./forlistanelkul.sh 1 2 3
A paraméterek száma: 3
Az egyes paraméterek: 1, 2, 3,
hallgato@ubuntu-server:~$
```

Az aktuális könyvtár minden állománynevét kisbetűsre változtatja

```
for f in *
do
   k=`echo $f | tr A-Z a-z`  # A nevet kisbetűsre
   if [ "$f" != "$k" ]       # Ha a név nem kisbetűs
   then
     mv "./$f" "./$k"   # Átnevez
   fi
done
```

Függvények használata

```
Függvénynév()
{
    utasítások
}

Függvényhívás
```

Egy ip intervallum végig pingelése, és a működő gépek kijelzése

```
mukodik_e()
{
  echo -n $1
  ping -c 1 $1 > /dev/null
  if [ $? -eq 0 ]
  then
    echo " működik."
  else
    echo " nem válaszol."
fi
}

clear
for i in 10.0.2.{1..255}
do
  mukodik_e $i
done
```

A Bash trap parancsa

```
bashtrap()
{
    echo "CTRL+C megnyomva!..."
}

trap bashtrap INT
clear
# Egy ciklus, hogy egyen idő kipróbálni
for i in `seq 1 10`
do
  sleep 5;
done
```

Adatbekérés a billentyűzetről

- echo "Add meg a neved : "
- read nev
- echo "Szia $nev, üdvözöllek!"

- # Készítsünk biztonsági másolatot a Dokumentumok könyvtárról
- # A névben legyen benne az időpont
- S=Mentes_$(date +%Y.%m.%d.%X).tar.gz
- tar -czf $S /home/hallgato/Dokumentumok

Aritmetika - műveletek

```
echo "Az első szám: "
read a
echo " A második szám: " & read b
echo "Egyszerű műveletek:"
# Az eredmény a "c" változóba kerül
let c=$a+$b
echo $a+$b=$c
let c=$a-$b
echo $a-$b=$c
let c=$a*$b
echo $a*$b=$c
let c=$a/$b
echo $a/$b=$c
let c=$a%$b
echo $a%$b=$c
let c=$a**$b
echo $a^$b=$c
```

Értékadó és módosító operátorok

- ((a = 23)) # Setting a value, C-style,
- #+ with spaces on both sides of the "=".
- echo "a (initial value) = $a" # 23
- ((a++)) # Post-increment 'a', C-style.
- echo "a (after a++) = $a" # 24
- ((a--)) # Post-decrement 'a', C-style.
- echo "a (after a--) = $a" # 23
- ((++a)) # Pre-increment 'a', C-style.
- echo "a (after ++a) = $a" # 24
- ((--a)) # Pre-decrement 'a', C-style.
- echo "a (after --a) = $a" # 23

- ((t = a<45?7:11)) # C-style trinary operator.

- echo "If a < 45, then t = 7, else t = 11." # a = 23
- echo "t = $t " # t = 7

Beépített változók

- $BASH - path to the Bash binary itself
- $EUID - "effective" user ID number
- $FUNCNAME - name of the current function
- $GROUPS - groups current user belongs to
- echo ${GROUPS[1]}
- $SECONDS - The number of seconds the script has been running
- $UID - User ID number
- $RANDOM : 0 - 32767 (előjeles 16-bites egész).

Számrendszer váltás

- szam=$((2#10101))
- echo $szam

- s=$((8#55))
- echo $szam

- szam=$((16#E1A))
- echo $szam

Tömb

- Csak egydimenziós
- Létrehozás:
  - area[11]=23
  - area[13]=37
  - area2=(egy kettő három négy öt)
  - base64_charset=({A..Z} {a..z} {0..9} + / =)
- Hivatkozás ({kapcsos zárójelek szükségesek})
  - echo -n "area[11] = " echo ${area[11]}
- Művelet
  - area[5]=`expr ${area[11]} + ${area[13]}`

Tömb

```
# A keresési útvonalakból egy tömböt csinál, aminek
# minden eleme egy elérési út
# A :-okat újsor jelekre cseréli
Tomb=$(echo $PATH | tr ":" "\n")
# Sorra veszi a tömb elemeit, és kiírja a képernyőre
for var in $Tomb
do
  echo $var
done
```

Sztring

- Karaktertömb elemeinek száma:
- ${#Sztring}
- Hivatkozás karaktertömb egy részletére
- ${Sztring:Kezdo:Hossz}
- Egy poziciótól mind
- ${Sztring:Kezdo}
- Behelyettesítés (első előfordulás)
- ${Sztring/Mit/Mire}
- Behelyettesítés (összes előfordulás)
- ${Sztring//Mit/Mire}

Sztring

- Törlés (legrövidebb illeszkedő részt)
- ${Sztring#Minta}
- Törlés (leghosszabb illeszkedő részt)
- ${Sztring##Minta}

s=xdcCF295CF

|---| legrövidebb

|----------| leghosszabb

echo ${s#c*F}

echo ${s##c*F}

Jelszógenerátor

```
#!/bin/bash
# jelszo.sh
echo "-------------------------------------"
echo "- Véletlen jelszó generálása -"
echo "-------------------------------------"
# Alapötlet: Antek Sawicki <tenox@tenox.tc>,
# A jelszóban használt karakterek
Karakterek="0123456789ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz"
# A jelszó hossza
Hossz="8"
# A jelszó karaktereit egyesével állítjuk elő
```

Jelszógenerátor

```
for i in `seq 1 $Hossz`
do
  # A Karakterek tömb elemeinek száma
  ElemSzam=${#Karakterek}
  # Véletlenszám 1..ElemSzam-1
  Veletlen=$(($RANDOM%$ElemSzam))
  # A jelszó soron következő karaktere
  UjKar=${Karakterek:$Veletlen:1}
  # Hozzáadás az eddigi jelszóhoz
  Jelszo="$Jelszo$UjKar"
  # Az egészet egyetlen utasításban is megoldhatjuk
  # Jelszo="$Jelszo${Karakterek:$(($RANDOM%${#Karakterek})):1}"
done
echo "$Jelszo"
```

Változók idézőjelek között

- Az idézőjel megakadályozza a speciális karakterek újraértelmezését és az elválasztást a whitespace karaktereknél

```
List="one two three"
for a in $List
do
   echo "$a"
done
# one
# two
# three
for a in "$List"
do
    echo "$a"
done
# one two three
```

Csoporttagságok

```
#!/bin/bash
# csoportok.sh
CsoportSzam=${#GROUPS}
echo "Csoportagságok száma: $CsoportSzam"
for g in `seq 1 $CsoportSzam`
do
  ((sgid=${GROUPS[$g]}))
  # Kiírja az egyes csoportazonosítókhoz tartozó csoportneveket
  awk -F':' -v gid="$sgid" '{ if ( $3 == gid ) {print $1} }' /etc/group
done
```

expr

- Concatenates and evaluates the arguments according to the operation given (arguments must be separated by spaces). Operations may be arithmetic, comparison, string, or logical.
- expr 3 + 5
- expr 5 * 3 # Ez a szorzás
- y=`expr $y + 1`
- b=`expr $x = $y` # Test equality.
- b=`expr $a \> 10`
- b=`expr $a \<= 3`

expr

- # length: length of string
- b=`expr  length $a`
- b=`expr index  $a  23` # Első előfordulás helye
- b=`expr  substr  $a  2  6` # Részsztring kivétele

Menü select-tel

```
PS3='Please enter your choice: '
options=("Option 1" "Option 2" "Option3" "Quit")
select opt in "${options[@]}"
do
  case $opt in
    "Option 1")  echo "you chose choice 1" ;;
    "Option 2") echo "you chose choice 2" ;;
    "Option 3") echo "you chose choice 3" ;;
    "Quit") break ;;
    *) echo invalid option;;
  esac
done
```