

Számítógép Architektúrák
BSc
8. Gyak
2023.11.22

Készítette:

Tóth Zsombor Gábor
Programtervező Informatikus BSc
D0H157

Miskolc, 2023

1. feladat – Készítsen egy shell script fájlt, mely bemeneti paraméterként beolvas két számot (kedvenc számai), és kiírja az:

- összegüket,
- különbségüket,
- szorzatukat,
- hányadosukat,
- osztási maradékukat a standard outputra.

./beolvas.sh 5 3

```
#!/bin/bash

# Ellenőrizzük, hogy legalább két bemeneti paramétert megadtak-e
if [ "$#" -lt 2 ]; then
    echo "Két számra van szükség!"
    exit 1
fi

# Bemeneti számok beolvasása
szam1=$1
szam2=$2

# Számokkal végzett műveletek
osszeg=$((szam1 + szam2))
kulonbseg=$((szam1 - szam2))
szorzat=$((szam1 * szam2))

# Ellenőrzés, hogy a második szám ne legyen 0, mert nem lehet nullával osztani
if [ "$szam2" -eq 0 ]; then
    echo "A második szám nem lehet 0, mert nem lehet nullával osztani."
    exit 1
fi

hányados=$((szam1 / szam2))
osztasi_maradek=$((szam1 % szam2))

# Eredmények kiírása
echo "Összeg: $osszeg"
echo "Különbség: $kulonbseg"
echo "Szorzat: $szorzat"
echo "Hányados: $hányados"
echo "Osztási maradék: $osztasi_maradek"
```

```
Összeg: 8
Különbség: 2
Szorzat: 15
Hányados: 1
Osztási maradék: 2
```

2. feladat – Készítsen egy my_script.sh fájlt, majd írja bele a kedvenc számát: favourite_number=? Jelenítse meg a következő szöveggel: A kedvenc számom:

```
#!/bin/bash

# Kedvenc szám beállítása
favourite_number=42

# Kedvenc szám kiírása
echo "A kedvenc számom: $favourite_number"
```

3. feladat – Készítsen egyszerű szkriptet, amely bekér a felhasználótól egy nevet és egy telefonszámot, majd a következő formátumban kiírja azokat egy neptunkod.json fájlba:

```
{ "name" : " a felhasználó által beírt név ",
  "phone" : " a felhasználó által beírt telefonszám "
}
```

```
#!/bin/bash

# Felhasználótól kérjük be a nevet
read -p "Kérem adja meg a nevet: " nev

# Felhasználótól kérjük be a telefonszámot
read -p "Kérem adja meg a telefonszámot: " telefon

# JSON adatok létrehozása a bekért adatokkal
json_data="{ \"name\" : \"$nev\", \"phone\" : \"$telefon\" }"

# JSON adatok kiírása a neptunkod.json fájlba
echo $json_data > neptunkod.json

echo "Az adatok el lettek mentve a neptunkod.json fájlba."
```

```
Kérem adja meg a nevet: Tóth Zsombor Gábor
Kérem adja meg a telefonszámot: +06706612511
Az adatok el lettek mentve a neptunkod.json fájlba.
```

```
{ "name" : "Tóth Zsombor Gábor", "phone" : "+06706612511" }
```

4. feladat – Hozzon létre egy shell script fájlt, amely egy paraméterként kapott txt fájlban a happy szó minden előfordulását nem gondoltam a vizsgaidőszakra szövegre cseréli, és elmenti az új szöveget egy out.txt fájlba. Az echo helyett használja a printf parancsot

```
#!/bin/bash



# Ellenőrizzük, hogy legalább egy bemeneti paramétert megadtak-e
if [ "$#" -lt 1 ]; then
    echo "Használat: $0 input.txt"
    exit 1
fi

# Bemeneti fájl neve a paraméterből
input_file="$1"

# Ellenőrizzük, hogy a fájl létezik-e
if [ ! -e "$input_file" ]; then
    echo "A megadott fájl nem létezik: $input_file"
    exit 1
fi

# Szövegcserét végzünk és az eredményt kiírjuk az out.txt fájlba
printf "$(\sed 's/happy/nem gondoltam a vizsgaidőszakra/g' "$input_file")" > out.txt

echo "A happy szó minden előfordulását lecseréltük a vizsgaidőszakra az out.txt fájlba."
```

| | |
|---|---|
| D0H157_1122 >  input.txt | D0H157_1122 >  out.txt |
| 1 nagyon happy | 1 nagyon nem gondoltam a vizsgaidőszakra |

5. feladat – Írjon egy scriptet, amely letölt a következő URL-ről egy file1.txt, majd kiírja belőle a valid email címeket egy emails.txt fájlba, aztán törli az eredetileg letöltött fájlt.

URL: <https://raw.githubusercontent.com/bbalage/BashExamples/master/assets/file1.txt>

```
#!/bin/bash

# Letöltjük a file1.txt fájlt
wget -O file1.txt https://raw.githubusercontent.com/bbalage/BashExamples/master/assets/file1.txt

# Kinyerjük a valid email címeket a file1.txt fájlból
grep -E -o '\b[A-Za-z0-9._%+-]+@[A-Za-z0-9.-]+\.[A-Z|a-z]{2,}\b' file1.txt > emails.txt

# Töröljük az eredetileg letöltött file1.txt fájlt
rm file1.txt

echo "Az eredeti file1.txt fájl törölve."
```

```
D0H157_1122 > 📄 emails.txt
1   Jupici@gmail.com
2   howard.wayland@citromail.hu
3   ubuntu@gmail.com
4   bestrapper@gmail.com
5   ET@gmail.com
6   harapos@gmail.com
7   bob@gmail.com
8   obelix@gmail.com
```

6. feladat – Kérjen be két koordinátát a felhasználótól! Ezek legyenek egy téglalap két átlolnes sarka. A téglalap oldalai párhuzamosak a koordináta tengelyekkel. Írja ki a téglalap területét! A koordináták nem lehetnek lebegőpontosak!

```
#!/bin/bash

# Funkció a téglalap területének kiszámolására
calculate_rectangle_area() {
    local x1=$1
    local y1=$2
    local x2=$3
    local y2=$4

    # A téglalap területének kiszámítása: terület = oldal1 * oldal2
    local side1=$((x2 - x1))
    local side2=$((y2 - y1))
    local area=$((side1 * side2))

    echo "A téglalap területe: $area"
}

# Felhasználótól bekérjük az első pont koordinátáit
read -p "Kérem adja meg az első pont x koordinátáját: " x1
read -p "Kérem adja meg az első pont y koordinátáját: " y1

# Felhasználótól bekérjük a második pont koordinátáit
read -p "Kérem adja meg a második pont x koordinátáját: " x2
read -p "Kérem adja meg a második pont y koordinátáját: " y2

# Ellenőrzés, hogy a koordináták érvényes egész számok-e
if ! [[ $x1 =~ ^[0-9]+$ && $y1 =~ ^[0-9]+$ && $x2 =~ ^[0-9]+$ && $y2 =~ ^[0-9]+$ ]]; then
    echo "Hiba: A koordinátáknak érvényes egész számoknak kell lenniük."
    exit 1
fi

# Terület kiszámítása és kiírása
calculate_rectangle_area "$x1" "$y1" "$x2" "$y2"
```

```
Kérem adja meg az első pont x koordinátáját: 2
Kérem adja meg az első pont y koordinátáját: 3
Kérem adja meg a második pont x koordinátáját: 4
Kérem adja meg a második pont y koordinátáját: 5
A téglalap területe: 4
```

Önálló Feladatok

1. feladat – Hozzunk létre egy shell script fájlt, amely egy konfigurációs fájlt generál nekünk YAML formátumban. Ez egy rendkívül egyszerű formátum, ami jelen esetben így fog kinézni:

username: first input

version: second input

site: third input

A shell kód kérje be az inputokat a felhasználótól, és hozza létre a config.yml fájlt az inputoknak megfelelően!

```
D0H157_1122 > create_config.sh
1  #!/bin/bash
2
3  # Felhasználótól bekérjűk a felhasználónév, verzió és webhely információkat
4  read -p "Kérem adja meg a felhasználónevet: " username
5  read -p "Kérem adja meg a verziót: " version
6  read -p "Kérem adja meg a webhelyet: " site
7
8  # Ellenőrizzük, hogy minden mezőt megadtak-e
9  if [ -z "$username" ] || [ -z "$version" ] || [ -z "$site" ]; then
10     echo "Hiba: Minden mezőt ki kell tölteni."
11     exit 1
12 fi
13
14 # Config.yml fájl létrehozása
15 cat <<EOF > config.yml
16 username: $username
17 version: $version
18 site: $site
19 EOF
20
21 echo "A config.yml fájl elkészült!"
```


2. feladat – Adott a korábbi órákról ismert people.csv fájl. Írjon egy olyan szkriptet, ami kicseréli benne a gmail-es email címeket citromail-esre, és kiírja az új tartalmat a neptunkod_people.csv fájlba! Megjegyzés: people1.csv használva a people.csv helyett, mivel az tartalmaz email nevű oszlopot, míg a people.csv nem.

```
D0H157_1122 > swap_email.sh
1  #!/bin/bash
2
3  # Kicseréljük a gmail-es e-mail címeket citromail-esre és
4  # kiírjuk a neptunkod_people.csv fájlba
5  awk -F',' 'BEGIN {OFS=","} {if ($1=="email") print; else
6  gsub(/@gmail\.com/, "@citromail.com", $1); print}' people1.
7  csv > neptunkod_people.csv
8
9  echo "Az e-mail címek cseréje megtörtént. Az eredményt a
10 neptunkod_people.csv fájlban találod."
```


3. feladat – Adott a korábbi órákról ismert people.csv fájl. Írjon egy olyan szkriptet, ami bemeneti paraméterként megkapja a keresett ember nevét, és kiírja az életkorát!

```
D0H157_1122 > find_age.sh
1  #!/bin/bash
2
3  # Bemeneti név
4  keresett_nev=$1
5
6  # Keresés a people.csv fájlban a bemeneti név alapján
7  szuletesi_datum=$(awk -F';' -v nev="$keresett_nev" '$1 ==
   nev {print $2}' people.csv)
8
9  # Ellenőrizzük, hogy talált-e egyezést
10 if [ -z "$szuletesi_datum" ]; then
11     echo "Nincs találat a(z) \"$keresett_nev\" nevű
      személyre a people.csv fájlban."
12 else
13     # Születési dátum formátumának ellenőrzése
14     if [[ $szuletesi_datum =~ ^[0-9]{4}\.[0-9]{2}\.[0-9]{2}
        \.$ ]]; then
15         # Kiszámoljuk az életkort
16         szuletesi_ev=$(echo "$szuletesi_datum" | cut -d'.'
            -f1)
17         jelenlegi_ev=$(date +%Y)
18         életkor=$((jelenlegi_ev - szuletesi_ev))
19
20         echo "$keresett_nev született: $szuletesi_datum,
            életkora: $életkor év."
21     else
22         echo "Hiba: Hibás születési dátum formátum a
            people.csv fájlban."
23         exit 1
24     fi
25 fi
26
```

```
Robert Bob született: 1997.09.12., életkora: 26 év.
```

Feltételes Operátorok

1. feladat – Készítsen egy shell scriptet, amely bemenetként egy téglalap két oldalának hosszát várja, és kiírja a síkidom területét! Valósítsa meg csak egész számokkal! (Természetesen végezzen ellenőrzéseket az inputon!)

```
D0H157_1122 > calc_area.sh
1  #!/bin/bash
2
3  # Funkció a téglalap területének kiszámolására
4  calculate_rectangle_area() {
5      local oldal1=$1
6      local oldal2=$2
7
8      # Terület kiszámítása: terület = oldal1 * oldal2
9      local terület=$((oldal1 * oldal2))
10
11     echo "A téglalap területe: $terület"
12 }
13
14 # Ellenőrizzük, hogy legalább két bemeneti paramétert
    megadtak-e
15 if [ "$#" -ne 2 ]; then
16     echo "Használat: $0 <oldal1> <oldal2>"
17     exit 1
18 fi
19
20 # Bemeneti oldalak
21 oldal1=$1
22 oldal2=$2
23
24 # Ellenőrizzük, hogy a bemeneti értékek egész számok-e
25 if ! [[ $oldal1 =~ ^[0-9]+$ && $oldal2 =~ ^[0-9]+$ ]]; then
26     echo "Hiba: Mindkét oldalnak egész számnak kell lennie."
27     "
28     exit 1
29 fi
30
31 # Téglalap területének kiszámítása és kiírása
32 calculate_rectangle_area "$oldal1" "$oldal2"
```

2. feladat – Adott egy fájl `nev_id_parok.txt` néven, ami id és név párosokat tartalmaz. Készítsen egy shell scriptet, ami bekéri a nevet, és kiírja a hozzá tartozó id-t, vagy hibát ad, ha a név nem található a fájlban.

```
D0H157_1122 > find_id.sh
1  #!/bin/bash
2
3  # Ellenőrizzük, hogy a nev_id_parok.txt fájl létezik-e
4  if [ ! -e "nev_id_parok.txt" ]; then
5      echo "Hiba: A nev_id_parok.txt fájl nem létezik."
6      exit 1
7  fi
8
9  # Felhasználótól bekérjük a nevet
10 read -p "Kérem adja meg a nevet: " keresett_nev
11
12 # Keresés a nev_id_parok.txt fájlban a bemeneti név alapján
13 id=$(grep "^$keresett_nev:" nev_id_parok.txt | cut -d':' -f2)
14
15 # Ellenőrizzük, hogy talált-e egyezést
16 if [ -z "$id" ]; then
17     echo "Nincs találat a(z) \"$keresett_nev\" névvel a nev_id_parok.txt fájlban."
18 else
19     echo "$keresett_nev id-je: $id"
20 fi
21
```

```
Kérem adja meg a nevet: zsuzso
zsuzso id-je: aef7421b
```

3. feladat – Az MVK Zrt. elérhetővé tesz egy szabványos GTFS adatbázist a fejlesztők számára, hogy menetrendi adatokat a saját applikációba tudják integrálni. Írjon egy shell script fájlt, amely letölti ezt az adatbázist, és kilistázza belőle azokat az utakat, amelyek a Centrumból indulnak, vagy a Centrumba ennek! Parancsok: wget, unzip (kitömörítésre), cat, grep Szükséges ellenőrzések: Ha a letöltendő fájl már egyszer le volt töltve, akkor az újbóli letöltés előtt töröljük az előző verziót! Ha egy mappába már korábban ki lett tömörítve a letöltött állomány, akkor az újbóli kitömörítés törölje mappa tartalmát!

```
D0H157_1122 >  gtfs_centrum.sh
1  #!/bin/bash
2
3  if [ -e gtfs.zip ]; then
4      rm gtfs.zip
5  fi
6
7  if [ -d gtfs ]; then
8      rm -r gtfs
9  fi
10
11  wget "https://gtfsapi.mvkzrt.hu/gtfs/gtfs.zip"
12  unzip gtfs.zip -d gtfs
13  cat gtfs/routes.txt | grep "Centrum"
14
```

4. feladat – Készítsen egy shell scriptet, ami bekéri a felhasználó születési dátumát yyyy.mm.dd formátumban! Ellenőrizze a dátum helyességét, és írja ki, hogy a felhasználó hány éves! Használja a date parancsot a jelenlegi dátum lekérésére!

```
D0H157_1122 >  életkor.sh
1   #!/bin/bash
2
3  calculate_age() {
4      IFS='.' read -r by bm bd <<< "$1"
5      IFS='.' read -r cy cm cd <<< "$(date +%Y.%m.%d)"
6
7      # Átváltás 10-es számrendszerbe
8      by=$((10#$by))
9      bm=$((10#$bm))
10     bd=$((10#$bd))
11     cy=$((10#$cy))
12     cm=$((10#$cm))
13     cd=$((10#$cd))
14
15     age=$((cy - by - (cm < bm || (cm == bm && cd < bd))))
16     echo "$age"
17 }
18
19 # Bekéri a felhasználó születési dátumát és ellenőrzi a
    helyességét
20 while true; do
21     read -p "Adja meg a születési dátumát 'ÉÉÉÉ.HH.NN'
        formátumban: " szuletesi_datum
22
23     if [[ ! $szuletesi_datum =~ ^[0-9]{4}\.[0-9]{2}\.[0-9]{2}$ ]]; then
24         echo "Hiba: Helytelen formátum. Kérjük, használja
            az 'ÉÉÉÉ.HH.NN' formátumot."
25     else
26         break
27     fi
28 done
29
30 életkor=$(calculate_age "$szuletesi_datum")
31
32 echo "Az életkora: $életkor év."
```


Önálló Feladatok

1. feladat – Valósítsa meg az 1. példa feladatát, de ezúttal lebegőpontos számokkal! Készítsen egy shell scriptet, ami bemenetként egy téglalap két oldalának hosszát várja, és kiírja a síkidom területét! (Természetesen végezzen ellenőrzéseket az inputon!)

```
D0H157_1122 > onallo_1.sh
1  #!/bin/bash
2
3  # Funkció a téglalap területének kiszámolására
4  calculate_rectangle_area() {
5      local oldal1=$1
6      local oldal2=$2
7
8      # Terület kiszámítása: terület = oldal1 * oldal2
9      local terület=$(echo "$oldal1 * $oldal2" | bc)
10
11     echo "A téglalap területe: $terület"
12 }
13
14 # Ellenőrizzük, hogy legalább két bemeneti paramétert
    megadtak-e
15 if [ "$#" -ne 2 ]; then
16     echo "Használat: $0 <oldal1> <oldal2>"
17     exit 1
18 fi
19
20 # Bemeneti oldalak
21 oldal1=$1
22 oldal2=$2
23
24 # Ellenőrizzük, hogy a bemeneti értékek számok-e
25 if ! [[ $oldal1 =~ ^[0-9.]+$ && $oldal2 =~ ^[0-9.]+$ ]];
    then
26     echo "Hiba: Mindkét oldalnak számnak kell lennie."
27     exit 1
28 fi
29
30 # Téglalap területének kiszámítása és kiírása
31 calculate_rectangle_area "$oldal1" "$oldal2"
32
```

2. feladat – Valósítsa meg a 2. példa feladatát, de ezúttal ne csak name_id_pairs.txt nevű fájlra működjön, hanem bármilyen nevű fájlra! A fájl nevét a script bemeneti paraméterként fogadja! Ellenőrizze, hogy a fájl létezik és olvasható-e, mielőtt a funkciók további részét megvalósítjuk!

```
D0H157_1122 > onallo_2.sh
1  |#!/bin/bash
2
3  # Ellenőrizzük, hogy legalább két bemeneti paramétert
   megadtak-e
4  if [ "$#" -ne 1 ]; then
5      echo "Használat: $0 <fájlnév>"
6      exit 1
7  fi
8
9  # Bemeneti fájl név
10 fajlnev=$1
11
12 # Ellenőrizzük, hogy a fájl létezik és olvasható-e
13 if [ ! -r "$fajlnev" ]; then
14     echo "Hiba: A fájl nem létezik vagy nem olvasható."
15     exit 1
16 fi
17
18 # Felhasználótól bekérjük a nevet
19 read -p "Kérem adja meg a nevet: " keresett_nev
20
21 # Keresés a bemeneti fájlban a bemeneti név alapján
22 id=$(grep "^$keresett_nev:" "$fajlnev" | cut -d':' -f2)
23
24 # Ellenőrizzük, hogy talált-e egyezést
25 if [ -z "$id" ]; then
26     echo "Nincs találat a(z) \"$keresett_nev\" névvel a
       $fajlnev fájlban."
27 else
28     echo "$keresett_nev id-je: $id"
29 fi
30
```


3. feladat – Valósítsa meg a 3. példa feladatát, de ezúttal a Centrum helyett bármelyik végállomást fogadja el, és bemeneti paraméterként adja át azt a scriptnek. Ha nincs ilyen végállomás, írjon hibaüzenetet!

```
D0H157_1122 > onallo_3.sh
1  #!/bin/bash
2
3  if [ -e gtfs.zip ]; then
4      rm gtfs.zip
5  fi
6
7  if [ -d gtfs ]; then
8      rm -r gtfs
9  fi
10
11  wget "https://gtfsapi.mvkzrt.hu/gtfs/gtfs.zip"
12  unzip gtfs.zip -d gtfs
13
14  vegallomas=$1
15  talalat=$(cat gtfs/routes.txt | grep "$vegallomas")
16
17  if [ -n "$talalat" ]; then
18      echo "$talalat"
19  else
20      echo "Hiba: Nincs találat!"
21  fi
```