PROGRAMOVÁNÍ V SHELLU

Prostředí, jazyk, zdrojový kód

- chceme-li posloupnost jistých příkazů používat opakovaně, případně z různých míst adresářové struktury, můžeme tuto posloupnost uložit do souboru, který necháme zpracovat interpretrem příkazů
- interpretr příkazů může být sh, bash nebo další
- cestu k interpretru příkazů zadáme v prvním řádku zdrojového kódu takto:

  #! /bin/bash
- souboru necháme bez přípony (Linux sám rozpozná typ souboru podle obsahu) nebo mu přidáme příponu .sh
- nezapomeňte po vytvoření souboru zkontrolovat práva (kdo jej chce spouštět, musí mít právo x)
- soubor spustíme voláním shellu s parametrem tvořeným jménem souboru

  sh jmeno_skriptu nebo jej necháme zpracovat aktuálním shellem (příkaz tečka)
  ./jmeno_skriptu

Proměnné

- Proměnné není třeba před použitím deklarovat
- Datový typ je přiřazen prvním použitím proměnné
- Při prvním použití zadáváme pouze jméno proměnné
- Při každém dalším použití se na proměnnou odkazujeme pomocí znaku dolar

Výpis na obrazovku

- Výpis základního stavu systému, není třeba definovat žádné proměnné

Př.:
```
#!/bin/bash
# Skript pro testovani stavu systemu

#Vypis aktuálního data a času
date

#Základní systémové informace
uname -a #(může být jen uname -nvr)

#Jak dlouho je systém „nahoře“
uptime

#Kdy byl systém rebootován
who -b

#Kdo byl naposledy přihlášen (posledních 15 přihlášení)
last | tail -15

#Stav síťových adaptérů
Ip link

#Testovací ping na www.unob.cz
ping -c 3 www.unob.cz
```

Tento skript můžeme nazvat např. stav (stav.sh) a pokud bychom ho chtěli spouštět zadáním jeho jména odkudkoliv, musíme ho umístit (nakopírovat) do některého adresáře ze systémových cest => výpis platných cest *echo $PATH*. Mezi tyto cesty patří i adresář bin v home directory uživatele -> (~/bin)
Pokud adresář ~/bin neexistuje a není jako cesta v $PATH -> vytvořit adresář a provést přidání cesty ~/bin do parametru *$PATH => PATH=$PATH:$HOME/bin*; potom *export PATH* (nebo přímo *export PATH=$HOME/bin:$PATH)* => platí pouze pro aktuální přihlášení (terminálové okno). V souboru .profile by měla být zadaná kontrola cesty $HOME/bin a pokud tento adresář existuje, pak je při přihlášení vyexportován do parametru PATH -> (kontrola obsahu ~/.profile)
Pokud by měl být spouštěn všemi uživateli, pak je třeba ho umístit do veřejné přístupného systémového adresáře (např. /usr/local/bin) -> nutno použít sudo a nastavit správná přístupová práva (755, root.root)

POZOR!! Pokud se skript nazývá stav.sh, musíme ho vyvolat příkazem stav.sh **=>** rozdíl oproti .exe, .com a .bat souborům systému Windows **<=** pro jednodušší vyvolání nebudeme používat koncovku

- Pomocí příkazu echo a uvozovek. Proměnné se přiřadí hodnota a ta se vypíše.

Př.:
```
#!/bin/bash

x=5
echo „Hodnota proměnné x je $x.“
```

Čtení z klávesnice

- Pomocí příkazu ***read promenna,*** čte vstup až do stisku klávesy enter, vše přiřadí do jedné proměnné
- Můžeme načíst současně více proměnných pomocí ***read promenna1 promenna2 promenna3,*** každý úsek oddělený mezerníkem se načte do nové proměnné

Př.:
```
#!/bin/bash

echo "Zadej hodnotu x:"
read x
echo „Hodnota proměnné x je $x."
```

Př.:
```
#!/bin/bash

echo "Zadej jmeno a prijmeni"
read jmeno prijmeni
echo „Uzivatel se jmenuje: $jmeno $prijmeni"
```

- Mezera odděluje jednotlivé proměnné. Pokud zadáme více proměnných, než je načítáno, do poslední deklarované proměnné se načte úsek od mezery před poslední deklarovanou proměnnou.

Př.:
```
#!/bin/bash

echo "Zadej jmeno a prijmeni"
# Zadame trislovne jmeno, napr. Bezny Franta Uzivatel
read jmeno prijmeni

echo „Uzivatel se jmenuje: $jmeno $prijmeni"
        # Vypis vypada uplne stejne, jako v předchozím případu
echo „Jmeno uzivatele je: $jmeno."
echo „Prijmeni uzivatel je: $prijmeni."
        # V proměnné $prijmeni jsou definovany poslední dva řetězce (Franta Uzivatel)
```

- Pokud zadáme méně proměnných, než je požadováno, do poslední deklarované proměnné se načte prázdný řetězec.

Př.:
```
#!/bin/bash

echo "Zadej jmeno, prijmeni a druhé prijmeni"
# Zadame dvouslovne jmeno, napr. Bezny Franta
read jmeno prijmeni druheprijmeni

echo „Uzivatel se jmenuje: $jmeno $prijmeni $druheprijmeni."
        # Vypis vypada uplne stejne, jako v předchozím případu
echo „Jmeno uzivatele je: $jmeno."
echo „Prijmeni uzivatel je: $prijmeni."
echo „Druhé prijmeni uzivatel je: $druheprijmeni."
        # V proměnné $druheprijmeni je prázdný řetězec
```

Uvozovky a apostrofy

- " " Normální uvozovky. Řetězec uvnitř těchto uvozovek je chápán jako obyčejný text, metaznaky jsou ignorovány. Do textu jsou vkládány hodnoty proměnných.
- ' ' Apostrof vedle klávesy enter. Řetězec uvnitř těchto apostrofů je chápán shellem jako obyčejný text, je zamezeno nahrazování proměnných jejich hodnotou.
- ` **příkaz**` Obrácené apostrofy. Řetězec je shellem chápán jako příkaz k vykonání; tento řetězec je vykonán před zpracováváním zbytku řádku a výsledek příkazu nahradí původní řetězec `retezec`. Novější ekvivalent těchto apostrofů je $(příkaz)

Př.:

```
#!/bin/bash

text=10; echo "text je $text;" # vytiskne: text je 10;
text=10; echo 'text je $text;' # vytiskne: text je $text;
text=whoami; echo `$text;` # vypise napr: sstetina
```

Logické výrazy

K vyhodnocení logického výrazu používáme konstrukci ***[testovany vyraz]*** (pozor na povinné mezery kolem závorek).

Testovat lze např.:

- Typ souboru
  - [–f soubor] existuje soubor a je obyčejným souborem?
  - [–d soubor] existuje soubor a je adresářem?
  - [–b soubor] existuje a je to blokový speciální soubor?
  - [–c soubor] existuje a je to znakový speciální soubor?
  - [–e soubor] existuje?
  - [–p soubor] existuje a je to pojmenovaná roura (FIFO)?
  - [–s soubor] existuje a má délu větší než 0?
  - [–S soubor] existuje a je to soket?
  - [–L soubor] existuje a je to symbolický odkaz?
- Práva k souboru
  - [–r soubor] můžu číst soubor?
  - [–w soubor] můžu zapisovat do souboru?
  - [–x soubor] mohu soubor spustit?
  - [–O soubor] vlastním soubor?
  - [–G soubor] vlastní skupina soubor?
  - [–N soubor] byl soubor od posledního čtení změněn?
- Řetězce
  - [řetězec1=řetězec2] jsou řetězce identické?
  - [řetězec1 != řetězec2] jsou řetězce rozdílné?
  - [–z řetězec] je délka řetězce nulová (je řetězec prázdný)?
  - [–n řetězec] je délka řetězce nenulová?
  - [řetězec1<řetězec2] je řetězec1 abecedně před řetězcem2?
  - [řetězec1>řetězec2] je řetězec1 abecedně za řetězcem2?
- Numerické testy
  - [číslo1 –eq číslo2] čísla 1 a 2 se rovnají
  - [číslo1 –ne číslo2] čísla se nerovnají
  - [číslo1 –lt číslo2] číslo 1 je menší než číslo 2
  - [číslo1 –le číslo2] číslo 1 je menší nebo rovno číslu 2
  - [číslo1 –gt číslo2] číslo 1 je větší než číslo 2
  - [číslo1 –ge číslo2] číslo 1 je větší nebo rovno číslu 2

Testování výrazu se často používá ve spojení s podmínkou (if [podmínka] then else fi, případně v cyklu while [podmínka] do).

Aritmetické výrazy

Vyhodnocení matematického výrazu se dá provést pomocí

- příkazu expr, spojeného s obrácenými apostrofy nebo příkazem $()

např. *x=$(expr $x + 1) nebo x=,expr $x + 1,*

- příkazu $(())

např. *#!/bin/sh*

x=8
y=4
z=$(($x + $y))
echo „Součet $x a $y je $z“

Řídicí konstrukce

Podmínka if

if výraz; then příkazy1; [elif příkazy2; then příkazy3;] ... [else příkazy4;] fi

Př.: *#!/bin/bash*

```
if grep -q student01 /etc/passwd
then
        echo Student01 tam je
else
        echo Student01 tam neni
fi
```

Výše uvedený příklad můžeme použit také pro zadání hledaného loginu

Př. *#!/bin/bash*

```
echo "Zadej login, který hledáš."
read hledejlogin
if grep -q $hledejlogin /etc/passwd
then
        echo "Login $hledejlogin tam je."
else
        echo "Login $hledejlogin tam není."
fi
```

Musíme jen dát pozor, že příkazem grep hledáme řetězec. Takže pokud napíšeme student, tak nám toto porovnání potvrdí pravdivost i v případě loginu student01, student02, …
Pro vyhledávání loginů v /etc/group je lepší napsat výše uvedený příklad s dvojtečkou u čtení z klávesnice, protože v /etc/passwd je právě dvojtečka oddělovačem mezi jednotlivými hodnotami.

```
read hledejlogin
if grep -q $hledejlogin: /etc/passwd
then                -> pokud napíši student, porovnávat se bude řetězec student:
```

Hledání souboru a následná práce s ním

Př.

```
#!/bin/bash

if [ -f /etc/group ]
then    #Soubor existuje. Zkopíruj tedy soubor
        cp /etc/group .
        echo „Hotovo!“
else    # Soubor neexistuje. Vypiš tedy chybu
        echo „Soubor neexistuje!“
exit
fi
```

I v tomto skriptu můžeme hledaný soubor zadat z klávesnice.

```
#!/bin/bash

echo “Zadej hledaný soubor v adresáři /etc, který chceš zkopírovat.“
read hledanysoubor
if [ -f /etc/$hledanysoubor ]
then    #Soubor existuje. Zkopíruj tedy soubor
        echo “Hledaný soubor $hledanysoubor v adresari /etc existuje a bude zkopirovan.“
        cp /etc/group .
        echo „Hotovo!“
else    # Soubor neexistuje. Vypiš tedy chybu
        echo „Soubor $hledanysoubor v adresáři /etc neexistuje!“
exit
fi
```

Na níže uvedeném obrázku je skript, kde zjišťujeme, zda existuje v /etc/passwd určitý login. Pokud ano, zjišťujeme, zda má založený domovský adresář /home/login. Pokud ne, adresář je založen (musí být použit příkaz sudo, vytváří se adresář v adresáři home) a jsou nastavena (také přes sudo) příslušná práva pro práci s domovským adresářem pro daného uživatele.

V obou případech (domovský adresář existuje nebo je vytvořen) se zjišťuje, zda v domovském adresáři jsou konfigurační soubory .profile, .bashrc a .bash_logout. Pokud nejsou, jsou zkopírovány z adresáře /etc/skel a jsou jim nastavena správná přístupová práva.˝

!!!! Je třeba si dát pozor, že první testovací podmínka (if) pro testování přítomnosti loginu je ukončena až na konci skriptu, kdy je při nepřítomnosti loginu vypsaná informace o neexistenci loginu, a pak je teprve podmínka ze začátku skriptu ukončena (fi) -> další podmínky jsou vnořené do podmínky první **!!!!**

```
#! /bin/bash

echo "Zadej login, ktery hledas:"
read hledejlogin
if grep -q $hledejlogin: /etc/passwd;
then
        echo "hledany login $hledejlogin tam je."
#else
#       echo "hledany login $hledejlogin tam neni."
#fi

# vyhledani, zda existuje domovsky adresar zadaneho loginu.

if [ -d /home/$hledejlogin ]
then
        echo "Domovsky adresar loginu $hledejlogin existuje."
else
        echo "Domovsky adresar loginu $hledejlogin neexistuje, bude vytvoren."
        sudo mkdir /home/$hledejlogin
        sudo chown $hledejlogin.Students /home/$hledejlogin
        echo "Domovsky adresar loginu $hledejlogin vytvoren."
fi

# vyhledani, zda existuje konfiguracni soubor .profile.

if [ -f /home/$hledejlogin/.profile ]
then
        echo "konfiguracni soubor .profile existuje."
else
        echo "konfiguracni soubor .profile neexistuje, bude zkopirovan."
        sudo cp /home/student01/.profile /home/$hledejlogin/
        sudo chown $hledejlogin.Students /home/$hledejlogin/.profile
        echo "konfiguracni soubor .profile zkopirovan a nastaveny prava."
fi

# vyhledani, zda existuje konfiguracni soubor .bashrc

if [ -f /home/$hledejlogin/.bashrc ]
then
        echo "konfiguracni soubor .bashrc existuje."
else
        echo "konfiguracni soubor .bashrc neexistuje, bude zkopirovan."
        sudo cp /home/student01/.bashrc /home/$hledejlogin/
        sudo chown $hledejlogin.Students /home/$hledejlogin/.bashrc
        echo "konfiguracni soubor .bashrc zkopirovan a nastaveny prava."
fi


else
        echo "hledany login $hledejlogin tam neni."
fi
                                                                    6,1           All
```

Vícenásobné větvení – case

case slovo; in

vzor1) příkazy;; vzor2) příkazy;;

****) příkazy;;***

esac

Př.:

```
#!/bin/bash

read –p “znak?“ znak
echo –n “znak $znak je:”

case $znak in
        [0-9]) echo cislo.;;
        [A-Z]) echo velke pismeno.;;
        [a-z]) echo male pismeno.;;
        *) echo jiny znak;;
esac
```

Cykly

Cyklus for:

- předem znám počet opakování

for promenna in seznam

do

prikazy

done

- prvky seznamu jsou odděleny mezerou nebo tabulátorem

Př.:

```
#!/bin/bash

for i in 1 3 5 7
do
        echo $i
done
```

- místo konkrétních hodnot může být na místě seznamu „žolík“ – zástupný znak, např. znak hvězdička:

```
#!/bin/sh

for i in `ls *.sh`
do
        echo $i
done
```

Tento cyklus můžeme využít pro přípravu skriptu pro založení uživatelů.

V seznamu prvků opakování uvedeme loginy, které chceme založit.

```
#! /bin/bash

# skript pro vytvoření uživatelských účtů, ze seznamu loginů ve skriptu uvedených

for i in student11 student12 student13 student14 student15
do
        sudo useradd -c $i -g Students -m -s /bin/bash $i
        echo “Založen účet s loginem 4i.“
        echo “Výpis ze souboru /etc/passwd:“
        echo `getent passwd $i`
        sleep 3
done
```

Skript, kdy je výčet dán určitou podmínkou opakování

Př.

```
#! /bin/bash

for (( a=1 ; $a-4 ; a=$a+1 ))
do echo $a
done
```

Nejprve je přiřazena hodnota 1, toto přiřazení se provede pouze na začátku prvního průchodu cyklem. Druhý výraz slouží jako rozhodnutí, zda se výraz bude vykonávat. V okamžiku, kdy hodnota $a-4 bude nulová, tak se již příkaz za podmínkou nevykoná.

Tento princip můžeme použít např. při zakládání účtů , kdy měníme číselnou řadu v loginu (student01, student02, …)

Zde si můžeme definovat počáteční i koncovou hodnotu, pro kterou budeme loginy zakládat.

Než budeme zakládat uživatele, otestujeme si funkčnost připočítávání čísla 1.

Př.

```
#! /bin/bash

echo “Zadej počáteční číslo zakládaných uživatelských účtů student:“
read x
echo “Zadej koncové číslo zakládaných uživatelských účtů student:“
read y

for (( a=$x; $a-($y+1); a=$a+1 ))
do
        echo $a
done
```

Pokud bychom používali tento skript, musíme zajistit, aby se pro řadu 1-9 použila dvoumístná čísla (01-09).

Potřebujeme tedy otestovat, zda je číslo < 10.

Ve skriptu, kde je již provedeno také založení uživatelských účtů *žák* by to vypadalo např. následovně.

```
#! /bin/bash

echo "Zadej pocateni cislo zakladanych uzivatelskych uctu zak:"
read x

echo "Zadej koncove cislo zakladanych uzivatelskych uctu zak:"
read y

for (( a=$x; $a-($y+1); a=$a+1 ))
do
        if (( $a < 10 ))
                then
                        echo 0$a
                        sudo useradd -c "Zak c.0$a" -u 600$a -g 100 -m -d /home/Za
ci/zak0$a -s /bin/bash zak0$a
                        echo "Zalozen ucet s loginem zak0$a."
                        echo "Vypis ze souboru /etc/passwd:"
                        echo `getent passwd zak0$a`
                        sleep 2
                else
                        echo $a
                        sudo useradd -c "Zak c.$a" -u 60$a -g 100 -m -d /home/Zaci
/zak$a -s /bin/bash zak$a
                        echo "Zalozen ucet s loginem zak$a."
                        echo "Vypis ze souboru /etc/passwd:"
                        echo `getent passwd zak$a`
                        sleep 2

        fi
done
~
```

Výše uvedený skript můžeme upravit tak, aby uzivatel u zakládaných účtů po spuštění skriptu zadal název zakládaných účtů, počáteční a koncové číslo a hodnotu tisícovky a stovky u UID.

```
#! /bin/bash

echo "Zadej nazev zakladanych uzivatelskych uctu:"
read l

echo "Zadej pocateni cislo zakladanych uzivatelskych uctu:"
read x

echo "Zadej koncove cislo zakladanych uzivatelskych uctu:"
read y

echo "Zadej hodnotu tisicovky a stovky pro UID zakladanych uzivatelskych uctu:"
read u

for (( a=$x; $a-($y+1); a=$a+1 ))
do
        if (( $a < 10 ))
                then
                        echo 0$a
                        echo $l'0'$a
                        if grep -q $l'0'$a: /etc/passwd
                        then
                                echo "Zadany login $l"0"$a jiz existuje!!!"
                                echo `getent passwd $l'0'$a`
                                sleep 1
                        else
                        sudo useradd -c "$l c.0$a" -u $u'0'$a -g 100 -m -d /home/$l-home/$l'0'$a -s /bin/bash $l'0'$a
                        echo "Zalozen ucet s loginem $l"0"$a."
                        echo "Vypis ze souboru /etc/passwd:"
                        echo `getent passwd $l'0'$a`
                        sleep 2
                        fi
                else
                        echo $a
                        if grep -q $l$a: /etc/passwd
                        then
                                echo "Zadany login $l$a jiz existuje!!!"
                                echo `getent passwd $l$a`
                                sleep 1
                        else
                                sudo useradd -c "$l c.$a" -u $u$a -g 100 -m -d /home/$l-home/$l$a -s /bin/bash $l$a
                                echo "Zalozen ucet s loginem $l$a."
                        echo "Vypis ze souboru /etc/passwd:"
                        echo `getent passwd $l$a`
                        sleep 2

                        fi
        fi
done
                                                                                3,9           All
```

Cyklus while

- **cyklus bude probíhat, dokud je podmínka splněna**

  while podminka do

  prikazy

  done

Př.

```
#! /bin/bash

# inicializace počítadla
x=1
# nastavení opakování 5x
while [ $x -le 5 ]
do
        # výpis v každém opakování
        echo “Opakuji $x krát!!!“
        # připočtení po 1
        x=$(( $x + 1 ))    nebo     (( x++ ))
done
```

Nekonečný cyklus -> ukončení pouze přes Ctrl+C

Př.

```
#! /bin/bash
while :
do
        echo “Stiskni <Ctrl+C> k ukončení.“
        Sleep 1
Done
```

Ukončení cyklu za určité podmínky.

Př.

```
#! /bin/bash

# inicializace počítadla
x=1
# nastavení opakování 10x
while [ $x -le 10 ]
do
        # kontrola hodnoty x
        If [ $x == 6 ]
        then
                echo “přerušeno“
                break
        fi
        # výpis v každém opakování
        echo “Opakuji $x krát!!!“
        # připočtení po 1
        x=$(( $x + 1 ))    nebo     (( x++ ))
done
```

Tento skript můžeme s malou úpravou použít pro výpis všech opakování, vyjma opakování, které testujeme

Př.

```
#! /bin/bash

# inicializace počítadla
x=0
# nastavení opakování 10x
while [ $x -le 10 ]
do
        # navýšení hodnoty o 1
        (( x++ ))

        # kontrola hodnoty x
        If [ $x == 6 ]
        then
                continue
        fi
        # výpis v každém opakování
        echo “Opakuji $x krát!!!“
done
```

Jedním z častých použití cyklu while je čtení ze souboru řádek po řádku.

Př.

```
#! /bin/bash

file=/etc/passwd

while read -r line
do
        echo $line
done < “$file“
```

V tomto případě nám přepínač -r zajišťuje, že v případě přečtení zpětného lomítka nebude toto v textu ignorováno.

Výše uvedený skript můžeme použít i se zadáním argumentu (soubor, který čteme) při spuštění a s testováním, zda byl argument zadán.

Př.

```
#! /bin/bash

# testování, zda byl argument při spouštění skriptu zadán
If [ $# -gt 0 ]
then
        file=$1
        while read -r line
        do
                echo $line
        done < “$file“
else
        # Zpráva, pokud argument chybí
        echo “Chybí název souboru pro čtení!!!“
fi
```

Skript můžeme také upravit, aby čtený soubor zadal uživatel po spuštění skriptu

Př.

```
#! /bin/bash

echo “Zadej plnou cestu k souboru:“
read file

while read -r line
do
        echo $line
done < “$file“
```

a varianta se testováním, zda zadaný soubor skutečně existuje

Př.

```
#! /bin/bash

echo “Zadej plnou cestu k souboru:“
read file

# testování, zda zadaný soubor skutečně exsituje
If [ -f $file ]
then
        while read -r line
        do
                echo $line
                sleep 0.4
        done < “$file“
else
        echo “Zadaný soubor neexistuje!!!“
fi
```

Varianta se čtení po řádcích nám umožňuje také načítat jednotlivé položky z daných řádků. Musíme pamatovat na to, že oddělovačem pro příkaz read je mezera nebo tabelátor.

➔ protiklady.txt

```
cerna    bila
svetlo   tma
pravda   nepravda
nahore   dole
daleko   blizko
maly     velky
den      noc
začátek  konec
zloděj   dobroděj
```

Př.

```
#! /bin/bash
while read tak opak
do
        echo "$tak – $opak"
        sleep 0.5
done < protiklady.txt
```

Tento způsob můžeme také použít při zautomatizovaném vytváření účtů z daného seznamu. Při vytváření seznamu pozor na mezery a tabelátory!!!

➔ Seznam01.txt

```
student21   Student   c.21   5021   Students   users,sudo
student22   Student   c.22   5022   Students   users
student23   Student   c.23   5023   Students   users,Fachmani
student24   Student   c.24   5024   Students   users,sudo,Fachmani
student25   Student   c.25   5025   Students   users
```

Př.

```
#! /bin/bash

while read login jmeno prijmeni uid primgroup groups

do

        echo "Založení uživatele $login s podrobnostmi $jmeno $prijmeni, UID=$uid, primární skupina $primgroup, další skupiny $groups."

        sudo useradd -c "$jmeno $prijmeni" -u $uid -g $primgroup -G $groups -m -s /bin/bash $login

        sleep 0.5

done < seznam01.txt
```

Do výše uvedeného příkladu zahrneme testování, zda login nebo uid už nejsou použity.

Př.

```
#! /bin/bash
while read login jmeno prijmeni uid primgroup groups
do
        if grep -q $login: /etc/passwd
        then
                echo “Login $login je jiz zalozen!!!“
                continue
        else
                if grep -q :$uid: /etc/passwd
                then
                        echo “Login s UID=$uid je jiz zalozen!!!“
                        continue
                else

        echo “Zakladani uživatele $login s podrobnostmi $jmeno $prijmeni, UID=$uid, primární skupina $primgroup, další skupiny $groups.“
        sudo useradd -c “$jmeno $prijmeni“ -u $uid -g $primgroup -G $groups -m -s /bin/bash $login
        sleep 0.5
        echo Hotovo!
                fi
        fi

done < seznam02.txt
```

Pro příkaz read můžeme použít parametr IFS (Internal Field separator), a tak už pro nás nejsou mezery nebo tabelátory limitujícím faktorem pro načítání jednotlivých položek na řádcích.

while IFS=“:“ read …

=> DÚ – vytvořit seznam svoji studijní skupiny s hodnostmi, jmény a příjmeními, ubytování a skript, který z daného seznamu načte jednotlivé hodnoty a založí účty.