EVROPSKÁ UNIE
Evropské strukturální a investiční fondy
Operační program Výzkum, vývoj a vzdělávání

MINISTERSTVO ŠKOLSTVÍ, MLÁDEŽE A TĚLOVÝCHOVY

jihomoravský kraj

OPERAČNÍ SYSTÉMY

Tvorba skriptů v unixových shellech

Metodický list

Auto: Ing. Marek Kocan, Metodik: Ing. Roman Koláčný

Recenzent: Mgr. Jiří Činčura

Rok vydání: 2023

Obsah

Cíle

Studenti se seznámí s teoretickými základy v oblasti linuxových – obecně pak unixových – shellů. Dále si osvojí základní dovednosti pro vytváření jednouchých skriptů v shellu Bash, a to na základě konkrétních příkladů předvedených vyučujícím i na základě samostatně vyzkoušených úkolů.

Student bude schopen vlastními slovy vysvětlit co je to linuxový shell, dokáže správně v přihlášeném systému identifikovat aktuální shell a zvládne vytvořit jednoduché skripty. Student dále dokáže vlastními slovy popsat základní programové konstrukce v shellu Bash. V dlouhodobém horizontu bude student schopen samostatně využít nové znalosti a dovednosti pro zjednodušení správy operačního systému, zejména pak s ohledem na možnosti automatizace pomocí skriptů.

Dovednosti

Student bude schopen vytvářet a spouštět skripty v shellu Bash (bash), a to včetně využití základních programových konstrukcí včetně vstupu, výstupu, podmínek a cyklů. Dále student zvládne zapracovat do skriptů či v rámci jejich spouštění využívat i další prvky linuxových operačních systémů. V dlouhodobém horizontu bude student schopen aplikovat získané dovednosti pro efektivní práci v linuxových operačních systémech.

Pracovní prostředí

Výuku lze realizovat v prostředí:

- Cylab JCEKB, operační systém Debian (linuxový server)

Pro práci postačí standardní nástroje, některé další budou v rámci průběhu scénáře nainstalovány.

1 Unixové shelly – teoretický základ

Unixové shelly jsou textovým uživatelským rozhraním pro práci s operačním systémem. Obdobou ve Windows je příkazový řádek CMD nebo PowerShell, obvykle pracuje i na úrovni vstupů a výstupů s prostým textem (na rozdíl například od PowerShellu, který pracuje s objekty). Textové rozhraní má co do kategorie ovládaní historickou vazbu na terminálový přístup k počítačům založeném na zasílání textových znaků a příjmu textových odpovědí (plus případně řídících informací, například pro posun či podbarvení nebo jinou formu zvýraznění).

Výhodou shellů je nízká náročnost na komunikaci a možnost využívat vestavěné programové konstrukce, tedy snadno vytvářet i velmi složité „programy“. Shell je ve skutečnosti jakousi obálkou pro příjem zadání, provedení požadavku a zajištění návratových dat. Uživatel nijak neřeší podstatu komunikace či provádění jednotlivých příkazů operačním systémem a shell tedy představuje abstrakci mezi počítačem a uživatelem (čímž umožňuje naplnit jednu z rolí operačního systému).

V unixovém, resp. i v linuxovém světě existuje široká škála shellů, přičemž mezi nejznámější patří bash (Bourne-Again SHell), sh (Bourne shell), ksh (Korn shell) či zsh (Z shell). Jednotlivé shelly se mírně odlišují co do svých možností, proto je nezbytné při tvorbě skriptů uvažovat nad kompatibilitou (za normu lze považovat podmnožinu Korn shellu a sh).

1.1 Bash

Jedním z nejoblíbenějších a nejrozšířenějších shellů je bash – jeho název vychází z Bourne-Again SHell a lze jej považovat za nástupce Bourne shellu (sh). Jde o shell podporující i složitější programové konstrukce včetně využívání řídících struktur. Bash využívá také proměnné, a to proměnné prostředí i lokální. Tyto konstrukce (mj. i skripty) formou interpreteru provádí v rámci kontextu, což je obvykle prostředí přihlášeného uživatele. Vytvořené skripty lze efektivně volat také automaticky.

Základní nápověda je dostupná v rámci manuálové stránky, `man bash`. Pro interaktivní práci s bash je potřeba přihlášení k systému pomocí terminálu, například putty.

2 Tvorba skriptů v unixových shellech – praktické ukázky a cvičení

V této části vyučující představí jednotlivé příklady související s tvorbou skriptů v linuxových – obecně pak v unixových – shellech. Na tyto ukázky bude průběžně navazovat samostatná práce studentů (kontrolní body).

2.1 Pracovní prostředí

Výuku lze realizovat v prostředí Cylab JCEKB – Linuxový server (Debian), bash.

2.2 Základní informace pro tvorbu skriptů

2.2.1 Jaký shell používám?

V návaznosti na konkrétní instalaci operačního systému a nastavení uživatele či jeho přihlášení může aktuálně přihlášený uživatel využívat různé shelly. S ohledem na rozdíly naznačené v teoretické části je proto při tvorbě skriptů důležité přesně znát, pro jaký shell budou skripty vytvářeny. *Vyučující by měl v tento okamžik naznačit, že v rámci jednoho shellu lze vytvářet skripty i pro jiný shell.*

Základním krokem pro zorientování se v prostředí je zjištění, jaké shelly jsou v rámci dané instalace k dispozici. K tomu lze využít konstrukci `cat /etc/shells`. Následující příklad ukazuje konkrétní využití.

```
$ cat /etc/shells
```

Výstup (dle skutečného obsahu):

```
$ /bin/sh
/bin/bash
/usr/bin/bash
/bin/rbash
/usr/bin/rbash
/bin/dash
/usr/bin/dash
```

Zjištění aktuálně využívaného shellu lze provést řadou různých způsobů, jednotlivé hlavní možnosti ukazující následující příklady.

```
$ ps -p $$
```

Výstup (dle skutečného obsahu):

```
$
    PID TTY          TIME CMD
    804 pts/0    00:00:00 bash
```

Vyučující by měl vysvětlit, k čemu slouží příkaz `ps` a kde studenti najdou název použitého shellu.

```
$ readlink /proc/$$/exe
```

Výstup (dle skutečného obsahu):

```
$ /usr/bin/bash
```

```
$ echo $0
```

Výstup (dle skutečného obsahu):

```
$ bash
```

V návaznosti na reakce třídy může vyučující uvést, že studenti se mohou v materiálech na internetu setkat i s konstrukcí `echo "$SHELL"`, *která ovšem může za určitých okolností vypsat jiný shell, než ve kterém se uživatel aktuálně nachází.*

Kontrolní bod

Studenti si po přihlášení k terminálu nejméně dvěma způsoby zjistí, v jakém shellu se aktuálně nachází.

2.2.2 Editace skriptů

Pro editaci skriptů lze využít v podstatě jakýkoli editor, s ohledem na obtížnost lze doporučit využití editoru nano, který má základní použití `nano <název_editovaného_souboru>`. Následující příkaz vyvolá editaci souboru s názvem `skript.bash`.

```
$ nano skript.bash
```

Vyučující předvede základní ovládání editoru nano a použití funkčních kláves Ctrl+O (uložení), Ctrl+W (vyhledávání) a Ctrl+X (ukončení).

Kontrolní bod

Studenti si v domovském adresáři založí soubor skript.bash s obsahem:

`TEST`

Následně potvrdí existenci tohoto souboru pomocí příkazu `ls` *a vypsáním jeho obsahu pomocí příkazu* `cat`*. Tuto druhou část kontrolního bodu zadá vyučující pouze za situace, kdy studenti uvedené příkazy již znají. Pokud neznají, řešení předvede.*

2.2.3 Skript č. 1 – úvod

V rámci prvního skriptu vyučující předvede jednoduché konstrukce:

- `#!/usr/bin/env bash` – informační řádek (tzv. shebang) předepisující, jaký shell má být pro spuštění, resp. interpretaci použitý. Vyučující dále upozorní na to, že se lze sekat i s dalšími obdobnými konstrukcemi – např. `#!/bin/bash` – které ovšem mohou být v případě nestandardní instalace či nestandardního umístění souborů nefunkční či problémové. Doporučovaná konstrukce vychází z proměnné prostředí a je tedy univerzální.
- `echo` – výpis parametru (v tomto skriptu textového řetězce) na standardní výstup
- `exit` – ukončení provádění skriptu. Tento příkaz není povinný, stejně tak se může vyskytovat bez parametru (pak je návratový kód provádění skriptu 0, tj. bez chyby).
- `#` – komentář

```
$ nano skript1.bash
```

V prostředí editoru nano bude vložen následující kód:

```
#!/usr/bin/env bash
echo "Ahoj"
exit 0
```

Následně dojde ke spuštění skriptu `skript1.bash` pomocí konstrukce (*vyučující upozorní na to, že nejde o jedinou možnost spuštění a že jednu z dalších ukáže v dalším kroku*):

```
$ bash skript1.bash
```

Výstup:

```
$ Ahoj
```

Dále bude kód prvního skriptu rozšířen o komentář – vše začínající znakem `#` je ignorováno až do konce řádku. (*Vyučující může zmínit, že komentáře se v této podobě často využívající například i v konfiguračních souborech*.)

```
$ nano skript1.bash
```

V prostředí editoru nano bude původní kód upraven na:

```
#!/usr/bin/env bash
echo "Ahoj"
# toto jsou komentáře
exit 0
```

Vyučující demonstruje, že při spuštění nedojde ke změně.

```
$ bash skript1.bash
```

Výstup:

```
$ Ahoj
```

Kontrolní bod

Studenti si vkládají do kódu skriptu `skript1.bash` prázdné řádky a diskutují s vyučujícím o tom, proč se nic při spuštění skriptu nezměnilo.

Další možností pro spouštění skriptů je nastavit soubor se skriptem jako spustitelný pomocí příkazu `chmod` a spustit jej jako běžný spustitelný soubor.

```
$ chmod 700 skript1.bash
$ ./skript1.bash
```

Výstup:

```
$ Ahoj
```

Vyučující dle potřeby vysvětlí příkaz chmod a upozorní na důležitost použití shebangu.

2.2.4 Skript č. 2 – proměnné

V rámci skriptu vyučující předvede jednoduché konstrukce:

- Přiřazení hodnoty do proměnné

```
$ nano skript2.bash
```

V prostředí editoru nano bude vložen následující kód:

```
#!/usr/bin/env bash

cesta=~/

echo "Vypisuji domácí adresář aktuálně přihlášeného uživatele"
echo "---Začátek výpisu---"
ls ${cesta} -al  # -a vše, -l podrobnosti
echo "---Konec výpisu---"
echo $cesta # kontrolní výpis proměnné
exit 0
```

`cesta=~/` – do proměnné s názvem cesta je přiřazena cesta k domovskému adresáři. V případě řádku `ls ${cesta} -al` je demonstrováno použití proměnné bezpečnou cestou – uzavření do složených závorek. (*Vyučující v případě potřeby vysvětlí parametry příkazu `ls` a upozorní na použití komentáře.*)

Následně dojde ke spuštění skriptu `skript2.bash` pomocí konstrukce:

```
$ bash skript2.bash
```

Výstup (dle skutečnosti):

```
$
Vypisuji domácí adresář aktuálně přihlášeného uživatele
---Začátek výpisu---
total 38
drwxr-xr-x 4 sysadmin sysadmin 4096 Jun 25 19:35 .
drwxr-xr-x 3 root     root     4096 Jun  4 16:31 ..
-rw------- 1 sysadmin sysadmin   10 Jun  4 16:41 .bash_history
-rw-r--r-- 1 sysadmin sysadmin  220 Jun  4 16:31 .bash_logout
-rw-r--r-- 1 sysadmin sysadmin 3526 Jun  4 16:31 .bashrc
drwxr-xr-x 3 sysadmin sysadmin 4096 Jul 28 19:35 .local
-rw-r--r-- 1 sysadmin sysadmin  807 Jul  4 16:31 .profile
drwxr-xr-x 2 sysadmin sysadmin 4096 Jul 27 21:57 skripty
---Konec výpisu---
/home/sysadmin/
```

Skripty lze v případě potřeby i „ladit“ pomocí zobrazování informací o aktuálně prováděném příkazu, čehož lze dosáhnout opět několika různými cestami, například pomocí přepínače `-x`:

```
$ bash -x skript2.bash
```

Výstup:

```
$
+ cesta=/home/sysadmin/
+ echo 'Vypisuji domácí adresář aktuálně přihlášeného uživatele'
Vypisuji domácí adresář aktuálně přihlášeného uživatele
+ echo '---Začátek výpisu---'
---Začátek výpisu---
+ ls /home/sysadmin/ -al
total 38
drwxr-xr-x 4 sysadmin sysadmin 4096 Jun 25 19:35 .
drwxr-xr-x 3 root     root     4096 Jun  4 16:31 ..
-rw------- 1 sysadmin sysadmin   10 Jun  4 16:41 .bash_history
-rw-r--r-- 1 sysadmin sysadmin  220 Jun  4 16:31 .bash_logout
-rw-r--r-- 1 sysadmin sysadmin 3526 Jun  4 16:31 .bashrc
drwxr-xr-x 3 sysadmin sysadmin 4096 Jul 28 19:35 .local
-rw-r--r-- 1 sysadmin sysadmin  807 Jul  4 16:31 .profile
drwxr-xr-x 2 sysadmin sysadmin 4096 Jul 27 21:57 skripty
+ echo '---Konec výpisu---'
---Konec výpisu---
+ echo /home/sysadmin/
/home/sysadmin/
+ exit 0
```

V rámci skriptu lze ladění povolovat a zakazovat pomocí konstrukcí `set -x` a `set +x`.

Kontrolní bod

Studenti si vyzkouší „ladění“ v případě skriptu `skript1.bash`

V rámci skriptů lze využít i přesměrování výstupu do souborů obdobně, jako by byly příkazy spouštěny přímo v prostředí shellu (terminálu), například:

```
#!/usr/bin/env bash

cesta=~/

echo "Vypisuji domácí adresář aktuálně přihlášeného uživatele"
echo "---Začátek výpisu---"
ls ${cesta} -al >vypis.out # -a vše, -l podrobnosti
echo "---Konec výpisu---"
echo $cesta # kontrolní výpis proměnné
exit 0
```

Ověření přesměrování výstupu příkazu `ls` je možné například pomocí příkazu `cat`:

```
$ cat vypis.out
```

Vyučující vysvětlí obsah výstupu včetně toho, proč v souboru vypis.out nejsou výstupy příkazů echo (muselo by být přesměrováno vše, například pomocí konstrukce `bash skript2.bash >vypis.out`). Dle potřeby může vyučující volání předvést nebo využít zadání pro samostatný či domácí úkol.

2.2.5 Skript č. 3 – vstup

V rámci skriptu vyučující předvede jednoduché konstrukce:

- `read` – interaktivní načtení proměnné

- `if` – podmínku (konstrukce je obdobná jako u jiných programovacích jazyků; míra vysvětlení záleží na okolnostech daných cílovou skupinou, záleží na volbě vyučujícího)

```
$ nano skript3.bash
```

V prostředí editoru nano bude vložen následující kód:

```
#!/usr/bin/env bash

echo "Zadej své jméno"
read jmeno
echo "Ty se opravdu jmenujes $jmeno?"
read odpoved
if [ "$odpoved" == "ANO" ]; then
    echo "Krásné jméno"
else
    echo "No tak vidíš!"
fi

exit 0
```

Následně dojde ke spuštění skriptu `skript2.bash` pomocí konstrukce:

```
$ bash skript3.bash
```

Vyučující diskutuje interaktivní vstup i výstup (při odpovědi ANO na druhou otázku skript vypíše jinou zprávu než při jakékoli další odlišné odpovědi). Dále vysvětlí, proč proměnná $odpoved je v uvozovkách (zabránění reinterpretace speciálních znaků a v tomto případě interpretace jako prostý textový řetězec).

Kontrolní bod

Studenti si vyzkouší vložení další otázky a opsání zadané odpovědi na standardní výstup

2.2.6 Skript č. 4 – cyklus while a for

V rámci skriptu vyučující předvede jednoduché konstrukce:

- `while` – cyklus s podmínkou vyhodnocovanou na začátku
- `for` – cyklus s opakováním dle číselné hodnoty

Vyučující upozorní na to, že použité konstrukce jsou pouze základními ukázkami a možnosti cyklů jsou v případě shellu Bash mnohem širší.

```
$ nano skript4.bash
```

V prostředí editoru nano bude vložen následující kód:

```
#!/usr/bin/env bash

while true; do
```

```
  uptime
  sleep 1
done

exit 0
```

Následně dojde ke spuštění skriptu `skript4.bash` pomocí konstrukce:

```
$ bash skript4.bash
```

Výstup:

```
$
 18:19:55 up 23:38,  2 users,  load average: 0.00, 0.00, 0.00
 18:19:56 up 23:38,  2 users,  load average: 0.00, 0.00, 0.01
 18:19:57 up 23:38,  2 users,  load average: 0.00, 0.00, 0.03
 18:19:58 up 23:38,  2 users,  load average: 0.00, 0.00, 0.10
```

Vyučující popíše příkaz `uptime` *(základní zobrazení aktuální vytížení) a* `sleep` *(přerušení provádění skriptu po zadanou dobu), diskutuje výstup a uvede způsob, jakým provádění použití nekonečného skriptu (podmínka true) ukončí – Ctrl+C. Cílem není detailní popis příkazu* `uptime`*, zde je použitý pouze jako příklad opakujícího se volání.*

```
$ nano skript5.bash
```

V prostředí editoru nano bude vložen následující kód:

```
#!/usr/bin/env bash

for (( a=1 ; $a-10 ; a=$a+1 ))
do echo $a
done

exit 0
```

Následně dojde ke spuštění skriptu `skript5.bash` pomocí konstrukce:

```
$ bash skript5.bash
```

Výstup:

```
$
1
2
3
4
5
6
7
8
9
```

Kontrolní bod

Studenti ve spolupráci s vyučujícím zkusí na základě svých dosavadních zkušeností, znalostí a dovedností vysvětlit parametry cyklu `for` *(počáteční hodnota 1, cyklus je prováděný, dokud je vyhodnocení výrazu* `$a-10` *nenulové, v každém dalším průchodu dojde k přičtení hodnoty 1 do proměnné a).*

Shrnutí a závěr

Studenti se seznámili jak se základními programovými konstrukcemi pro tvorbu skriptů v shellu Bash, tak i různými způsoby jejich volání. Dle svého uvážení může vyučující připravit pro studenty například test, a to jak z pohledu teorie, tak i praktického opakování nad zdokumentovanými postupy, případně zadat za domácí úkol tvorbu jednoduchého skriptu například na následující témata:

- vytvořte skript, který desetkrát vypíše text zadaný uživatelem (náročnost 4/10)
- vytvořte skript, který vypíše obsah adresáře v cestě zadané uživatelem (náročnost 6/10)
- vytvořte skript, který bude v nekonečné smyčce vypisovat text napevno zadaný v rámci skriptu (náročnost 3/10)

Seznam použitých zdrojů

SPI. *Manuálové stránky*. Dostupné z: https://manpages.debian.org