ZÁPADOČESKÁ
UNIVERZITA
V PLZNI

Semestrální práce

KIV/OS

Správce virtuálních strojů

Jiří Novotný A09N0032P
Zdeněk Janda A09N0076P
Miroslav Hauser A09N0037P

1 Zadání

Navrhněte a implementujte model správce virtuálních strojů. Shell tohoto správce bude obsahovat tyto příkazy: *cat, cd, echo, exit, kill, kshell, ls, man, ps, pwd, shutdown, sort*. Shell bude dále uchovávat aktuální pracovní adresář a historii použitých příkazů. K ovládání správce použijte vhodné rozhraní uživatel/stroj.
Doporučený postup:

1. Napište si terminálové okno.
2. Navrhněte gramatiku pro shell.
3. Navrhněte a implementujte syntaktický analyzátor.
4. Navrhněte strukturu správce virtuálních strojů.
5. Udělejte propojení na souborový systém, napište váš cat a ukažte jeho funkčnost.
6. Doplňte model o vykonávání příkazů shellu v hierarchii virtuálních strojů - init, login, shell, program xyz, ...
7. Přidejte přesměrování a roury.
8. Napište další programy pro příkazy shellu, logout, shell, date, ...
9. Model můžete rozšířit o send a receive mezi uživateli, vykonávání v pozadí, příkazy bg a fg.
10. K historii příkazů uchovávejte i adresáře, ve kterých byly použity
11. Dokončete detaily.
12. Testujte.

Během práce dodržujte tyto konvence:

- identifikátory a komentáře se píší anglicky
- názvy package pouze malými písmeny
- class soubory se generují do jiného adresáře (ne k zdrojákům)
- místo třídy kolekce se používá rozhraní

2 Analýza úlohy

3 Popis implementace

3.1 Konzole

Konzole se vytváří jako objekt třídy *UserInterface*, která je oděděna od třídy *JFrame*. Tento frame obsahuje *JTextAreu* jako jediný prvek. Jedná se o základní vstup a výstup správce virtuálních strojů. V rámci konstruktoru jsou nastaveny globální proměnné, vzhled okna konzole, namapování akcí a jako poslední je spuštěn první shell. Globální proměnné představují již zmíněnou *JTextAreu*, dále hodnotu offsetu od začátku textu *TAOff* a *lineHead*, což je univerzální začátek každého řádku konzole tzv. line header. Namapování jednotlivých akcí je nejobsáhlejší část vytváření konzole a obsahuje jak reakce na klávesnici, tak na myš. Podrobnému popisu se věnuje podkapitola o mapování akcí viz dále. Spuštění shellu obstarává metoda *createShell()*, která bude rozebrána v další části dokumentace.

3.1.1 Reakce na klávesnici

K namapování akcí klávesnice je použita metoda *consoleKeyActions(keyEvt)*. Ta v sobě obsahuje switch na jednotlivé kódy stisklých kláves (*getKeyCode()*). Vyhodnocované klávesy jsou: *ENTER, LEFT, RIGHT, UP, DOWN, PAGE UP, PAGE DOWN, HOME, TAB a BACKSPACE*. Klávesa *ENTER* umožnuje zpracování aktuální řádky zadané do konzole ať už shellem nebo právě běžící aplikací. Šipky slouží k práci s historii příkazů a pohybu v zadaném textu. Klávesa *HOME* je pozměněna tak, aby v případě běhu shellu skákala pouze na konec vypsaneho textu (za line header), tedy na hodnotu *TAOff*. V ostatních případech skáče až na začátek aktualního řádku. Klávesy *PAGE UP* a *PAGE DOWN* jsou zakázány úplně, aby nebylo možné skočit do již zpracované oblasti konzole. Klávesa *TAB* doplňuje rozepsaný příkaz resp. cestu v zadávaném řádku a klávesa *BACKSPACE* má opět přidáno omezení, aby nebylo možné mazat zpracované řádky konzole. Poslední akcí klávesnice je tzv. KeyStroke, tedy kombinace více kláves (klávesová zkratka). Je implementována reakce pouze **CTRL+D** určená k zaslání signálu odebrání konzole a ukončení aktualního procesu.

3.1.2 Reakce na myš

Odchytává se ve dvou případech:

1. **Kliknutí na X okna** - Při kliknutí na zavírací X konzole se volá metoda UserInterface.close(), která nejprve odhlásí uživatele a poté bezpečně ukončí všechny procesy a konzoli.

2. **Kliknutí nebo označení textu v konzoli** - Pokud se uživatel pokusí kliknout jinam než na konec konzole (aktuální řádek) nebo v případě označení textu je potřeba zamezit změně pozice kurzoru a tím i možnosti psaní mimo aktuální řádek. Oba problémy jsou řešeny metodou *mouseReleased(MouseEvent evt)*. Tedy kdykoli je v konzoli uvolněno tlačítko myší (ať po kliknutí nebo po označení) je kurzor nastaven na aktuální řádek (na konec textu konzole, který dán hodnotou offsetu *TAOff*).

3.2 Gramatika

3.2.1 Návrh

Návrh gramatiky byl odvozen od LINUXu. Pro zápis návrhu jsou použity tyto regulární výrazy:

- (x)? - žádný nebo jeden výskyt
- (x)* - žádný nebo několik výskytů
- xyz - pravidlo
- XYZ - lexikální výraz

Následně bude návrh gramatiky vypadat takto:

```
parse   ->  line (bg)?
line    ->  first ('<' in)? (next)* ('>' out)?
first   ->  cmd
next    ->  '|' cmd
cmd     ->  par (args)*
args    ->  par
in      ->  par
out     ->  par
bg      ->  '&'
par     ->  CHAR | STRING | ichar
ichar   ->  ICHAR
```

Na první pohled se může zdát návrch gramatiky zbytečně složitý, ale takto detailní rozložení je dále použito při návrhu lexikálního resp. syntaktického analyzéru. Oba analýzéry jsou generovány pomocí nástroje ANTLR 3.2, který

podle návrhu gramatiky vytvoří kód v Javě. Ten je navíc doplněn obslužným kódem pro jednotlivá pravidla tak, aby práci se zpracovanou řádkou co nejvíce ulehčil (proto je návrh tak podrobný). Celý návrh gramatiky s plnými pravidly pro lexikální i syntaktický analyzátor lze nalézt v souboru s gramatikou (OSVM_grammar.g).

3.2.2 Lexikální analyzér

Pro lexikální analyzér jsou definovány základní řídící znaky (roura, přesměrování, ad.) a mnoužina povolených symbolů pro názvy příkazů resp. souborů (CHAR, STRING). Dále je definován i doplněk obou množin ICHAR. Veškeré bíle znaky a znaky z množiny ICHAR jsou přeskakovány. Použité množiny:

- CHAR
  ('a'..'z' |'A'..'Z' |'0'..'9' |'/' |'_' |'-' |'?' |'.' |'..' |':')
- STRING
  CHAR*
- ICHAR
  !(' ' |'|' |'<' |'>' |'\n' |'&')

3.2.3 Syntaktický analyzér

Syntaktický analyzér rozdělí řádek do struktury *ArrayList*, který v sobě obsahuje další *ArrayList<String>*. V něm jsou uloženy jednotlivé argumenty příkazu a jako poslední položka je vždy název příkazu. Navíc ukládá cestu k vstupnímu resp. výstupnímu souboru do String proměnné *in* resp. *out* a poskytuje k nim přístup pomocí metod *getIn()* a *getOut()*. V poslední řadě nastavuje vlajky *bg* (pokud je příkaz spuštěn na pozadí) a *invalid* (pokud řádka obsahuje nějaký symbol z množiny ICHAR). Ty jsou po zpracování řádky přístupné přes metody *isBackgrounded()* a *containsInvalid()*.

3.3 Struktura VMM

3.3.1 bla

3.4 Vykonávání příkazů

3.4.1 bla

3.5 Přesměrování a roury

3.5.1 bla

3.6 Příkazy

3.6.1 bla

4 Uživatelská příručka

4.1 Překlad

Program lze snadno přeložit pomocí:

```
X:\app_path\>compile.bat
```

a spustit:

```
X:\app_path\>run.bat
```

4.2 Ovládání programu

Ovládání správce je intuitivní. Po spuštění (viz předchozí část) se objeví okno přihlášení. Po zadání uživatelského jména a potvrzení dojde k přihlášení uživatele k virtuálnímu stroji. Zde je možné používat základní příkazy shellu, včetně jejich argumentů a přesměrování vstupů resp. výstupů. Implementované příkazy: *cat, cd, echo, exit, kill, kshell, ls, man, ps, pwd, shutdown, sort.* Více k jednolivým příkazům v následující části.

4.2.1 cat

Použití: *cat [file]* [<ifile] [>ofile]*
Umožňuje vypsání žádného, jednoho nebo více souborů podle použití. Pokud se spustí cat bez argumenů nebo vstupu, pak používá vstup z konzole. Bez přesměrování výstupu vypisuje zadané řádky, v opačném případě je zapisuje do zadaného souboru a je možné ho využít jako jednoduchý textový editor. Ukončení se provádí odebráním konzole procesu pomocí zkratky *CTRL+D*

Pokud se spustí s argumenty, pak za sebe vypíše jednotlivé soubory v zadaném pořadí. Poslední vypsaný je soubor z přesměrovaného vstupu. Cat nelze použít *cat file >file* nebo *cat <file >file*, kde je stejný vstupní a výstupní soubor.

4.2.2 cd

Použití: *cd ..* nebo *cd rel_path* nebo *cd abs_path*
Umožňuje změnit pracovní adresář.

4.2.3 echo

Použití: *echo [arg]**
Vypíše řádek odpovídající zadaným argumentům.

4.2.4 exit

Použití: *exit*
Ukončí stávající shell. Pokud se jedná o poslední běžící shell, pak provede odhlášení uživatele a ukončení konzole.

4.2.5 kill

Použití: *kill pid*
Ukončí proces se zadaným PID.

4.2.6 kshell

Použití: *kshell*
Spustí další shell.

4.2.7 ls

Použití: *ls* nebo *ls rel_path* nebo *ls abs_path*
Vypíše aktualní resp. zadaný adresář. Lze použít zástupný příkaz *dir*.

4.2.8 man

Použití: *man*
Vypíše stručnou nápovedu ke správci strojů, která zahrnuje i popis jednotlivých příkazů.

4.2.9 ps

Použití: *ps* nebo *ps -u*
Vypíše všechny resp. uživatelovy běžící procesy.

4.2.10 pwd

Použití: *pwd*
Vypíše aktuální pracovní adresář.

4.2.11 shutdown

Použití: *shutdown*
Ukončí celý správce virtuálních strojů.

4.2.12 sort

Použití: *sort [file]* [<ifile] [>ofile]*
Umožňuje seřazení řádků vstupu. Pokud je spuštěn bez argumentů (s i bez přesměrování výstupu), pak načítá řádky z konzole. Po odebrání konzole (pomocí zkratky *CTRL+D*) vypíše na výstup seřazenou posloupnost zadaných řádků. Při spuštění s argumenty postupně načte všechny řádky všech zadaných souborů (včetně souboru z přesměrovaného vstupu) a poté vypíše jejich seřazenou posloupnost na výstup.

5 Závěr

Dokumentace byla vygenerována pomocí nástroje TEX.

6 Literatura

- **autor** – *kniha*, Cambridge University Press, Cambridge, 2003