UNIVERSIDADE DE BRASÍLIA - UNB - FGA
ENGENHARIA DE SOFTWARE
FUNDAMENTOS DE COMPILADORES

PROJETO DE COMPILADOR

PORTUGUÊS SIMPLIFICADO - SHELL SCRIPT

Guilherme de Lima Bernardes
Nilton Cesar C. Araruna
Gabriel Augusto Barbosa

Brasília, Agosto de 2013

PTShell

Manual

Sumário

1 - O PTShell

Programar em shell faz voltarmos a filosofia clássica do UNIX que é quebrar projetos complexos em subtarefas mais simples. Um conhecimento em shell script é essencial para qualquer um que deseja se tornar um administrador de sistemas, pois usando scripts, podemos realizar tarefas de administração muito mais rapidamente e muito mais facilmente (imagine a diferença entre ter que cadastrar trezentos usuários manualmente ou fazer um script com 6 linhas que cadastra todos estes usuários e ainda pode definir uma senha padrão para eles).

O principal objetivo do projeto é proporcionar uma alternativa para usuários mais leigos que precisam utilizar sistemas operacionais UNIX mesmo sem conter o conhecimento necessário. Isto será feito através de um compilador que seja capaz de traduzir uma linguagem natural, português com uma gramática mais enxuta e regras bem definidas, em uma linguagem de programação, shell script.

2 - Objetivo

Construir um compilador que transforme determinada sequência de comandos escritos em português simplificado para um shell script correspondente. O primeiro passo para atingir este objetivo é antes de mais nada definir claramente a linguagem fonte que será processada pelo compilador. Neste contexto será criada uma pseudo linguagem composta por um português simplificado, na qual seja possível representar comandos executados em ambiente UNIX. Esta abordagem assemelha-se ao Portugol (pseudocódigo escrito em português), entretanto com um nível de abstração maior.

Através desta solução um usuário com pouca experiência em shell será capaz de construir orações simplificadas na voz ativa que expressam as ações a serem realizadas, como por exemplo, remoção de arquivos, encerramento de processos, etc. O compilador a ser desenvolvido neste projeto receberá este código fonte e deverá gerar um shell script como código objeto, para que possa ser executado diretamente pelo usuário.

A sintaxe exata e as palavras chaves deste português simplificado serão definidas

de acordo com alguns comandos existentes no shell. Para que o escopo possa ser fechado corretamente alguns serão escolhidos e em seguida será realizado um mapeamento entre estes comandos e a necessidade do uso da linguagem natural para representá-los. Portanto a pseudo linguagem a ser utilizada como código fonte pelo compilador dará suporte apenas para os comandos definidos no escopo.

Comando UNIX	Funcionalidade
Sed	Substituir e “casar” padrões por meio de Expressões Regulares.
Grep	Filtragem textual por palavra chave
Mkdir	Criar diretório
Ls	Listar arquivos do diretório
Rm	Remover arquivo
Cd	Mudar diretório
Cp	Copiar arquivo ou diretório
Cat	imprimir arquivo na saída
Touch	criar arquivo

Tabela 1. Lista de comandos escolhidos.

Além da escolha dos comandos a serem incorporados a pseudo linguagem é também necessário realizar uma análise sobre as possíves combinações entre eles, verificando a complexidade envolvida nestas decisões, uma vez que o shell permite a concatenação de comandos (pipe - |).

3 - Pré Requisitos

- **Sistema operacional Linux.**

  Linux é um termo utilizado para se referir a sistemas operativos (português europeu) ou sistemas operacionais (português brasileiro) que utilizem o núcleo Linux. O núcleo Linux foi desenvolvido pelo programador finlandês Linus Torvalds, inspirado no sistema Minix. O seu código fonte está disponível sob a licença GPL (versão 2) para que qualquer pessoa o possa utilizar, estudar, modificar e distribuir livremente de acordo com os termos da licença.

- **Postgree SQL**

  O PostgreSQL é um poderoso sistema gerenciador de banco de dados objeto-relacional de código aberto. Tem mais de 15 anos de desenvolvimento ativo e uma arquitetura que comprovadamente ganhou forte reputação de confiabilidade, integridade de dados e conformidade a padrões. Roda em todos os grandes sistemas operacionais, incluindo GNU/Linux, Unix (AIX, BSD, HP-UX, SGI IRIX, Mac OS X, Solaris, Tru64), e MS Windows. É totalmente compatível com ACID, tem suporte completo a chaves estrangeiras, junções (JOINs), visões, gatilhos e procedimentos armazenados (em múltiplas linguagens). Inclui a maior parte dos tipos de dados do ISO SQL:1999, incluindo INTEGER, NUMERIC, BOOLEAN, CHAR, VARCHAR, DATE, INTERVAL, e TIMESTAMP. Suporta também o armazenamento de objetos binários, incluindo figuras, sons ou vídeos. Possui interfaces nativas de programação para C/C++, Java, .Net, Perl, Python, Ruby, Tcl, ODBC, entre outros, e uma excepcional documentação.

- **pgAdmin3**

O pgAdmin é um programa de administração Open Source que é uma plataforma de desenvolvimento para o PostgreSQL. O aplicativo pode ser usado em Linux, FreeBSD, Solaris, Mac OSX e plataformas Windows para gerenciar o PostgreSQL 7.3 e acima, executado em qualquer plataforma, assim como versões comerciais e derivados do PostgreSQL.

O pgAdmin é projetado para atender às necessidades de todos os usuários, desde escrever consultas SQL simples para o desenvolvimento de bancos de dados complexos. A interface gráfica suporta todas as funcionalidades do PostgreSQL e torna a administração mais fácil. O aplicativo também inclui um editor com destaque de sintaxe SQL, um editor de código do lado do servidor, um SQL / batch / shell agente de agendamento de trabalho.

O pgAdmin é desenvolvido por uma comunidade de especialistas do PostgreSQL em todo o mundo e está disponível em mais de uma dúzia de idiomas. É um Software Livre liberado sob a licença PostgreSQL .

- **MakeFile**

O objetivo de Makefile é definir regras de compilação para projetos de software. Tais regras são definidas em arquivo chamado Makefile. O programa make interpreta o conteúdo do Makefile e executa as regras lá definidas. Alguns Sistemas Operacionais trazem programas similares ao make, tais como gmake, nmake, tmake, etc. O programa make pode variar de um sistema a outro pois não faz parte de nenhuma normalização .

O texto contido em um Makefile é usado para a compilação, ligação(linking), montagem de arquivos de projeto entre outras tarefas como limpeza de arquivos temporários, execução de comandos, etc.

Vantagens do uso do Makefile:

- Evita a compilação de arquivos desnecessários. Por exemplo, se seu programa utiliza 120 bibliotecas e você altera apenas uma, o make descobre (comparando as datas de alteração dos arquivos fontes com as dos arquivos anteriormente compilados) qual arquivo foi alterado e compila apenas a biblioteca necessária.

- Automatiza tarefas rotineiras como limpeza de vários arquivos criados temporariamente na compilação

- Pode ser usado como linguagem geral de script embora seja mais usado para compilação

- **Flex**

O FLEX serve para gerar automaticamente programas (usualmente em “C”) fazendo a leitura
de uma entrada, de modo a varrer um texto e/ou programa (“scanners”) a fim de obter uma seqüência de unidades léxicas (“tokens”). Os tokens gerados pelos programas criados peloFLEX será usualmente processado posteriormente por um programa que realizará a analise sintática.
Lex => Gerador de analisadores léxicos (UNIX – Ex.: Lex AT&T, Berkeley BSD)
Flex => Gerador de analisadores léxicos (LINUX / Windows-DOS – GNU Lex)

Entrada: Arquivo de descrição do analisador léxico
Saída: Programa na linguagem “C” que realiza a análise léxica (default: lexyy.c)

- **Bison**

Bison é um gerador de interpretadores para fins gerais que converte uma gramática livre de contexto anotado em um LR determinista ou generalizada LR (GLR) parser empregando LALR (1) tabelas do analisador. Como um recurso

experimental, Bison também pode gerar IELR (1) ou canônica LR (1) tabelas do analisador. Uma vez que você esteja acostumado com Bison, você pode usá-lo para desenvolver uma vasta gama de analisadores de linguagem, daqueles usados em simples calculadoras de mesa até linguagens de programação complexas.

Bison tem compatibilidade ascendente com Yacc: todas as gramáticas válidas para Yacc devem funcionar com Bison sem alterações. Qualquer pessoa familiarizada com Yacc deve ser capaz de usar Bison com pouca dificuldade. Você precisa ser fluente em C ou C + + de programação para usar Bison. Java também é suportado como um recurso experimental.

4 - Repositório

Utilizamos a ferramenta GitHub para armazenarmos nosso repositório com o código do nosso projeto, que está sobre licença GPL versão 2.

Link do repositório do compilador PTShell:
https://github.com/guilhermedelima/compiler-pt-shell/

5 - Estrutura do Compilador PTShell

- bin: Diretório que contém o arquivo compiler que é o executavel do projeto.
- include: Diretório com as bibliotecas necessárias para o projeto.
- obj: Diretório que o makefile coloca os arquivos objetos da compilação do programa.
- resources: Diretório que contem os scripts para gerar o banco do projeto.
- src: Diretório com o código fonte do projeto.
- Makefile: Arquivo necessário para instalação e desinstalação do PTShell.

6 - Instalação e Desinstalação

- Na pasta include existe um arquivo chamado pt_spell.h onde a variavel POSTGRESQL_DB deve ser setada de acordo com as configurações do banco de dados instalado.

- Existe uma pasta chamada resources dentro da estrutura do compilador onde se encontra um arquivo do tipo sql que serve para criação do schema e das tabelas do banco de dados necessários para o compilador funcionar. Basta executar o arquivo sysntax_schema.sql no programa pgAdmin3.

- Abra o Terminal e navegue até o diretório do programa, que vem nomeado como compiler-pt-shell, utilizando o camando $cd <diretório>.

- digite o comando ‘make’ no terminal e aguarde a compilação de todos os arquivos necessários.
  obs: caso queira desinstalar é só utilizar o comando ‘make clean’ no lugar de ‘make’.

7 - Inicializando o PTShell

Modos de Uso

Dinâmico

Se o objetivo for apenas traduzir os comandos em tempo de execução, execute o programa compiler que está na pasta bin utilizando o terminal.

Exemplo do comando:

$ ./bin/compiler

Covencional

Crie um arquivo de texto com os comandos aceitos pelo PTShell que você deseja compilar, aconselhamos que seja criado dentro da pasta resources do PTShell, em

seguida é preciso abrir o terminal e navegar até a pasta compiler-pt-shell e então para compilar seu código é necessário digitar um comando que constitui de três partes:

1ª parte: chamar o compiler que é o executavel do compilador que está localizado dentro da pasta bin. Para executar o compiler use o comando “./bin/compiler”

2ª parte: caminho com o arquivo de texto que você criou para ser compilado, por exemplo “resources/seu_arquivo.txt”.

3ª parte: caminho no qual o arquivo com o código compilado será gerado e o nome do arquivo que irá conter o código, por exemplo “bin/codigo_gerado.txt”.

Exemplo do comando:

$ ./bin/compiler resources/seu_arquivo.txt bin/codigo_gerado.txt

8 - Regras definidas para utilizar os comandos

- **Palavras reservadas**
    - **Arquivo**: palavra reservada para chamada do arquivo que será utilizados pelos comandos suportados pelo compilador.
    - **para**: palavra reservada que indicará o diretório ou arquivo utilizados pelos comandos suportados pelo compilador.
    - **e:** palavra reservada para chamada do comando pipe.
    - **por:** palavra reservada que indicará frase utilizada pelos comandos suportados pelo compilador.
    - **no, nos:** palavra reservada que indicará o diretório ou arquivo utilizados pelos comandos suportados pelo compilador.
    - **diretório**: palavra reservada para chamada do diretório que será utilizados pelos comandos suportados pelo compilador.

- **frase**: palavra reservada que indicará frase utilizada pelos comandos suportados pelo compilador.

- **Anterior**: palavra reservada que servirá para voltar para um diretório anterior quando necessário.

- **Comando Ls**

O comando ls no Linux é usado para listar diretórios e arquivos. Se for executado o comando ls sem nenhum parâmetro vai ser listado todos arquivos e diretórios da pasta corrente. Para usá-lo basta digitar ls.

verbos aceitos: mostrar, exibir e ver.

exemplos:

- mostar DIRETORIO <nome do diretório>
- exibir ARQUIVO <nome do arquivo>

- **Comando Cat**

O comando cat exibe o que há dentro de determinado arquivo. Ele é útil quando deseja ler ou exibir um arquivo de texto.
Exemplo: cat TEXTO.txt – Exibe o conteúdo do arquivo TEXTO.txt

verbos aceitos: mostrar, exibir e ver.

Exemplos:

- mostrar ARQUIVO <nome do arquivo>

- **Comando RM**

O comando tem a função de remover arquivos. Tome cuidado ao utilizá-lo, pois caso você remova algum arquivo por engano, o erro será irreversível.
Exemplo: rm /home/baixaki/Arquivo.txt – O arquivo Arquivo.txt localizado na pasta /home/baixaki foi deletado.

verbos aceitos: Remover, apagar.

Exemplos:

- apagar ARQUIVO <nome do arquivo>
- apagar DIRETORIO <nome do diretório>

- **Comando MKDIR**

O comando mkdir cria diretórios.
Exemplo: mkdir DIRETORIO – A pasta DIRETORIO foi criada no local onde o usuário se encontrava.

verbos aceitos: criar, gerar

Exemplos:

- criar DIRETORIOS <nome do diretorio> <nome do diretorio>
- gerar DIRETORIOS <nome do diretorio> <nome do diretorio>

- **Comando Touch**

O comando Touch cria arquivos.

Exemplo: touch Arquivo – O arquivo foi criado no local onde o usuário se encontrava.

verbos aceitos: criar, gerar

Exemplos:

- criar ARQUIVO <nome do arquivo> <nome do arquivo>
- gerar ARQUIVO <nome do arquivo> <nome do arquivo>

- **Comando CD**

O comando “cd” serve para acessar e mudar de diretório corrente. Ele é utilizado para a navegação entre as pastas do computador.

Exemplo: cd /home/baixaki/Desktop – Acessa a pasta correspondente à área de trabalho do usuário baixaki.

verbos aceitos: mudar, ir, voltar.

Exemplos:

- mudar PARA DIRETORIO <nome do diretório>
- mudar PARA DIRETORIO ANTERIOR

- **Comando CP**

O comando cp copia arquivos – o famoso CTRL+C + CTRL+V.

Exemplo: $ cp Exemplo.doc /home/baixaki/Trabalho/EXEMPLO.doc - O arquivo EXEMPLO.doc foi copiado para a pasta /home/baixaki/Trabalho com o mesmo nome.

verbos aceitos: Copiar

Exemplos:

- copiar ARQUIVO <nome do arquivo> PARA <nome do arquivo ou nome do diretorio>
- copiar ARQUIVO <nome do arquivo> PARA DIRETORIO <nome do diretorio>
- copiar ARQUIVO <nome do arquivo> PARA ARQUIVO <nome do arquivo>
- copiar DIRETORIO <nome do diretorio> PARA <nome do diretorio>
- copiar DIRETORIO <nome do diretorio> PARA DIRETORIO <nome do diretorio>
- copiar ARQUIVOS <nome do arquivo> <nome do arquivo> PARA DIRETORIO <nome do diretorio>
- copiar DIRETORIOS <nome do diretorio> <nome do diretorio> PARA DIRETORIO <nome do diretorio>

- **Comando GREP**

O comando grep pode ser visto como uma forma simplificada de consulta a um banco de dados que consiste de texto puro, em que cada linha representa um registro. Pode ser usado para retirar um conjunto de strings (cadeias de caracteres) do resultado de um comando dado ou de um arquivo texto, por mais longo que seja. Os exemplos que vou dar aqui falarão por si.

Exemplo: grep -i net

verbos: encontrar, buscar e filtrar.

Exemplos:

- buscar FRASE “frase” no ARQUIVO <nome do arquivo>
- encontrar FRASE “frase” no ARQUIVO <nome do arquivo>

- **Comando SED**

O comando sed é uma das principais linguagens para manipulação de arquivos e streams do Unix/Linux. Com o sed é possível substituir e “casar” padrões, sempre por meio de Expressões Regulares

Exemplo: sed 'expressão' arquivo

verbos: substituir, trocar

Exemplos:

- substituir FRASE “frase” por “frase” no ARQUIVO <nome do arquivo>