[MAC0211] Laboratório de Programação I
Aula 18
Autoconf
Expressões Regulares

Alair Pereira do Lago

DCC–IME–USP

12 de maio de 2015

Na aula passada...

- Pré-processadores, pré-processadores léxicos, o Pré-processador do C (cpp)
- Macros, processadores de macros, processadores de macros de propósito geral
- GNU M4

GNU Autoconf

- Ferramenta usada na geração de pacotes de software portáteis
- Produz scripts *shell* que automaticamente configuram pacotes de código fonte de programas, para adaptá-los a diferentes tipos de SOs derivados do Unix
- Com isso, um programa pode ser compilado em dezenas de tipos diferentes de SOs, com milhares de configurações diferentes
- Os scripts verificam a presença de cada uma das dependências (ferramentas, bibliotecas, etc.) do programa
- Os scripts gerados pelo Autoconf combinam comandos *shell* e comandos do M4

GNU Autoconf – funcionamento

- A abordagem do Autoconf para conseguir um alto grau de portabilidade é verificar a presença de funcionalidades, ao invés de verificar as versões das ferramentas
- A saída do Autoconf é um programa de configuração que é colocado junto do seu código-fonte
- Seu código-fonte deve estar preparado, fazendo testes sobre macros especiais e adaptando-se às diversas situações de um modo padronizado
- Você também precisará do Mafekile.in, um makefile com macros, que será transformado no Makefile final, na hora de compilar

Autoconf – arquivos gerados

O Autoconf produz um script de configuração chamado `configure`, que, quando executado, cria vários arquivos, substituindo parâmetros de configuração existentes neles pelos valores apropriados.
Arquivos gerados pelo `configure`:

- um ou mais arquivos Makefiles
- [opcional] um arquivo de cabeçalho C (geralmente chamado de `config.h`), contendo diretivas `#define` (macros) que podem ser usadas nos fontes do seu programa
- um script *shell* chamado `config.status`, que quando executado recria os arquivos acima
- um arquivo de log *config.log*, que registra as mensagens produzidas pelos compiladores, auxiliando na depuração

Criação do configure

Um dos possíveis roteiros mais completos:

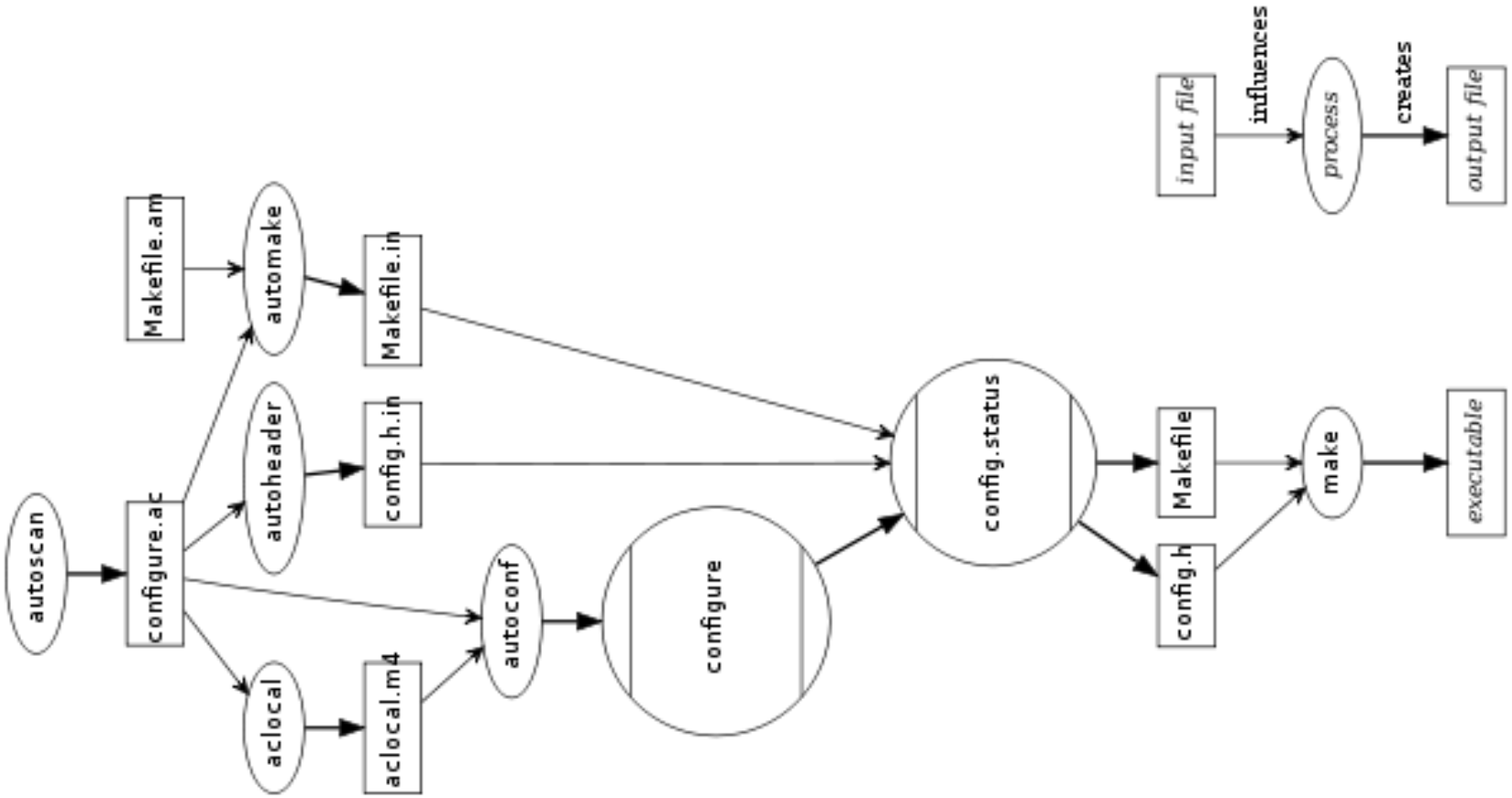
1. Rode o programa `autoscan` em seus códigos-fonte. A saída será um arquivo `configure.scan`
2. Faça ajustes manuais no `configure.scan` se necessário e salve o resultado em `configure.ac`
3. Rode o programa `autoheader`, que gerará o `config.h.in` (template para o `config.h`)
4. Rode o `autoconf`, que gerará o `configure`

Obs.: Tanto o `autoheader` quanto o `autoconf` usam o arquivo `configure.ac` como entrada.

O arquivo `configure`

- Códigos-fonte + `configure` + `Makefile.in` + `config.h.in` = conjunto (pacote) do programa
- Quando o `configure` é executado, ele faz todos os testes necessários e, em caso de sucesso, constrói os arquivos `Makefile` e o `config.h` a partir de seus "templates" `Makefile.in` e `config.h.in`
- Da mesma forma que podemos gerar um template para `config.h` usando o `autoheader`, também podemos gerar templates para Makefiles com a ferramenta GNU Automake.

Esquema para o Autoconf



Fonte: Wikipedia

Mudando de assunto...

Expressão regular

- Mini-linguagem declarativa usada para *descrever padrões de texto* (= especificar um conjunto de cadeias de caracteres)
- Geralmente, está incorporada em outras mini-linguagens
- É onipresente – é usada em muitas ferramentas e linguagens
- Substitui grandes volumes de códigos de implementação de diferentes funcionalidades de busca

Um exemplo simples de ferramenta que usa expressões regulares é o `grep`, um filtro que deixa passar para sua saída padrão toda linha de sua entrada que casa (*match*) com um padrão especificado.

Tipos no UNIX e seus derivados – I

- **Expressões *Glob*** – limitam-se às expressões envolvendo metacaracteres usadas nos shells antigos para o casamento com nomes de arquivos.
  Metacaracteres existentes:
  - `*` casa com qualquer sequência de caracteres.
    Exemplo: carn* casa com *carne*, *carneiro*, *carnudo*
  - `?` casa com exatamente 1 caracter simples.
    Exemplo: ?at casa com *Bat* ou *cat*, mas não com *at* ou *habitat*
  - `[...]` casa com uma classe de caracteres.
    Exemplo: [CB]at casa com *Cat* ou *Bat*, mas não com *cat*, *bat* ou *at*

Tipos no UNIX e seus derivados – II

- **Expressões regulares básicas** – notação aceita pelos utilitários `grep`, `sed` (stream editor) e `ed` (line editor)
- **Expressões regulares estendidas** – notação aceita pela versão estendida do `grep`, o `egrep`. Expressões regulares no *Lex* e no *Emacs* são muito parecidas com essas
- **Expressões regulares Perl** – notação aceita para as funções de expressões regulares de Perl e Python. Essas expressões são bem mais poderosas que as anteriores

Metacaracteres das expressões regulares básicas

.	um caracter qualquer
[abcd]	um caracter no conjunto dado
[a-zA-Z]	um caracter nos intervalos dados
[^abcd]	um caracter que não está no conjunto dado
^	início da cadeia de caracteres
$	fim da cadeia de caracteres. Em ferramentas que se baseiam no processamento de linhas, casa com o final da linha
*	elemento precedente zero ou mais vezes

Exemplos de expressões regulares básicas

'x.y'	x seguido por qualquer caracter seguido por y
'x\.y'	x seguido por um ponto seguido por y
'xz*y'	x seguido por qualquer número de instâncias de z, seguido por y. Ou seja, "xy" ou "xzy" ou "xzzzy", mas não "xz" nem "xdy"
"s[xyz]t"	s seguido por qualquer um dos caracteres x, y ou z, seguido por t. Ou seja, "sxt", "syt" ou "szt", mas não "st" nem "sat"
's[^x0-9]t'	s seguido por qualquer caracter que não seja x nem algum outro no intervalo 0–9, seguido por t. Ou seja, "slt" ou "smt", mas não "sxt", "s0t" nem "s4t"
'^x'	x no início de uma cadeia de caracteres. Ou seja, "xzy" ou "xzzy", mas não "yzy" nem "yxy"
'x$'	x no fim de um cadeia de caracteres. Então, "yzx" ou "yx", mas não "yxz" or "zxy"

Metacaracteres das expressões regulares estendidas

?	zero ou um caracter qualquer
+	elemento precedente uma ou mais vezes
(n1|n2|n3)	qualquer uma das opões listadas
{m,n}	elemento precedente pelo menos m e não mais que n vezes

Exemplos

"xz?y"	x seguido por no máximo um z seguido por y. Ou seja, "xy" ou "xzy", mas não "xz" nem "xdy"
"xz+y"	x seguido por uma ou mais instâncias de z, seguido por y. Ou seja, "xzy" ou "xzzy", mas não "xy" nem "xz"e nem "xdy"
"xz{2,5}y"	x seguido por 2, 3, 4 ou 5 instâncias de z, seguido por y

Algumas classes de caracteres POSIX

POSIX	ASCII	Descrição
[:alnum:]	[A-Za-z0-9]	caracteres alfa-numéricos
[:alpha:]	[A-Za-z]	caracteres do alfabeto
[:blank:]	[\t]	espaço e tabulação
[:cntrl:]	[\x00-\x1F \x7F]	caracteres de controle
[:digit:]	[0-9]	dígitos
[:lower:]	[a-z]	letras minúsculas
[:space:]	[\t \r \n \v \f]	caracteres brancos
[:upper:]	[A-Z]	letras maiúsculas
[:xdigit:]	[A-Fa-f0-9]	dígitos hexadecimais

Exemplos no *bash*

- Lista arquivos cujo nome começa com uma letra entre 'a' e 'h':
  ls -l [a-h]*
- Mostra as linhas de `arq.txt` que começam com a sigla de uma disciplina do MAC:
  grep -E '^MAC[[:digit:]]{3,4}' arq.txt
- Verifica se a sequência de caracteres digitada é um *username* válido (ou seja, que começa com letra, mas na sequência pode conter letras ou números, e possui tamanho mínimo de 2 caracteres e máximo de 7):
  grep -E '^[A-z][A-z0-9]{2,7}'

Obs.: A opção '-E' no `grep` habilita o uso de expressões regulares estendidas.

Bibliografia e materiais recomendados

- GNU Autoconf
  `http://www.gnu.org/software/autoconf/`
- Livro: *The Art of Unix Programming*, de Eric S. Raymond. Capítulo 8, Seção "Case Study: Regular Expressions"
  `http://www.catb.org/esr/writings/taoup/html/ch08s02.html#regexps`
- Livro: *Mastering Regular Expressions*, de Jeffrey Friedl, considerado a "Bíblia" das expressões regulares
  `http://regex.info/`
- GNU AWK – Guia do Usuário
  `http://www.gnu.org/software/gawk/manual/gawk.html`
- *An Awk primer* (tutorial de Awk bastante interessante)
  `http://www.vectorsite.net/tsawk.html`
- Notas das aulas de MAC0211 de 2010, feitas pelo Prof. Kon
  `http://www.ime.usp.br/~kon/MAC211`

Cenas dos próximos capítulos...

Na próxima aula:

- Geradores de analisadores léxicos
- GNU Flex