Exam 2 Review

# Notes From Instructor

Know how to email from shell

Program 3

Write make from scratch

Program 4

Implicit rules

Lex/Flex

All definitions of FLEX/BISION

Write an awk program to calculate a series of columns or rows.

Call from c++ file

Regex

Design REGEX to give result

Determine what results yield from a REGEX

# Main Points

## Make

### Purpose

A utility that automatically builds executable programs and libraries from source code

Manages automatically updating files when others change

Uses a provides control file (the makefile) to, find all modified code, re-compile to updated object files, re-link the object files to executable, maintain compiler flags and environments

### Syntax

<target> : [<dependency>]\*

[<TAB><command><endl>]+

\*zero or more

+1 or more

Ex:

myClass1.o: myClass1.cc myclass1.h

g++ -c myClass1.cc

### Rules

Implicit

Make will find a customary method for updating a target file if there is no explicit rule

Explicit

Rules from the makefile

Example

foo: foo.o bar.o

cc –o foo foo.o bar.o $(CXXFLAGS) $(LDFLAGS)

There is an explicit rule for the foo file

There is an implicit rule for foo.o and bar.o

Patterns

Contains the ‘%’, this rule is for all files matching the .o which has all .c for dependents

%.o:%.c

$(CC) –c $(CFLAGS) $(CPPFLAGS) $< -O $@

### Targets

Dummy target

Target that is not a file, like this one

clean:

\rm \*.o

### Dependencies

### Patterns

### Touch

### Compiler Options

## Awk

Interpreted programming language for text procession typically used as a data extraction and reporting tool

Takes input file and splits it into LINES.

Awk tests each line against condition, then does action on matches.

### **Conditions**

Default condition matches every line

### **Actions**

Default action prints line to stdout

### Shell scripts with arguments

BEGIN and END

Allow certain actions to take place before the stream is processed.

## Pipes

### Popen

## AutoConf

### Configuration Script

### Make, Make Test, Make Install

Creates a configuration shell script

Executing script performs operations on the OS.

A Makefile is generated that is specific to the OS.

#### Produces shell scripts

to configure (not build) software

#### Adaptions

Detecting compiler

Testing for existence of header files

Testing for existence of libraries

Performing custom tests

Configuring for compatibility

## SCM

Software Configuration Management, revision control, version control, source control

RCS/CVS

Git

Subversion

## Regex

Sequence of characters that forms a search pattern

Regular Language

Any language that can be expressed as a regular expression

#### Common Expressions

Empty Set - ∅

Empty String - no characters

Literal character - set containing only one character

#### Common Operations

Concatenation - combining strings

* {"ab", "c"}{"d", "ef"} = {"abd", "abef", "cd", "cef"}.

Alternation - set union

* R {"ab", "c"} and S {"ab", "d", "ef"}
* expression R | S describes {"ab", "c", "d", "ef"}.

Kleene Start \* - smallest superset of ε, and is closed under string concatenation

* {"ab", "c"}\* = {ε, "ab", "c", "abab", "abc", "cab", "cc", "ababab", "abcab", … }.

#### Delimiters

#### Standards

|  |  |
| --- | --- |
| **Metacharacter** | **Description** |
| **^** | Matches the starting position within the string. In line-based tools, it matches the starting position of any line. |
| **.** | Matches any single character (many applications exclude [newlines](https://en.wikipedia.org/wiki/Newline), and exactly which characters are considered newlines is flavor-, character-encoding-, and platform-specific, but it is safe to assume that the line feed character is included). Within POSIX bracket expressions, the dot character matches a literal dot. For example, a.c matches "abc", etc., but [a.c]matches only "a", ".", or "c". |
| **[ ]** | A bracket expression. Matches a single character that is contained within the brackets. For example, [abc] matches "a", "b", or "c". [a-z] specifies a range which matches any lowercase letter from "a" to "z". These forms can be mixed: [abcx-z] matches "a", "b", "c", "x", "y", or "z", as does [a-cx-z].  The - character is treated as a literal character if it is the last or the first (after the ^, if present) character within the brackets: [abc-], [-abc]. Note that backslash escapes are not allowed. The ] character can be included in a bracket expression if it is the first (after the ^) character: []abc]. |
| **[^ ]** | Matches a single character that is not contained within the brackets. For example, [^abc] matches any character other than "a", "b", or "c". [^a-z]matches any single character that is not a lowercase letter from "a" to "z". Likewise, literal characters and ranges can be mixed. |
| **$** | Matches the ending position of the string or the position just before a string-ending newline. In line-based tools, it matches the ending position of any line. |
| **( )** | Defines a marked subexpression. The string matched within the parentheses can be recalled later (see the next entry, \*n*). A marked subexpression is also called a block or capturing group. **BRE mode requires \( \)**. |
| **\*n*** | Matches what the *n*th marked subexpression matched, where *n* is a digit from 1 to 9. This construct is vaguely defined in the POSIX.2 standard. Some tools allow referencing more than nine capturing groups. |
| **\*** | Matches the preceding element zero or more times. For example, ab\*cmatches "ac", "abc", "abbbc", etc. [xyz]\* matches "", "x", "y", "z", "zx", "zyx", "xyzzy", and so on. (ab)\* matches "", "ab", "abab", "ababab", and so on. |
| **{*m*,*n*}** | Matches the preceding element at least *m* and not more than *n* times. For example, a{3,5} matches only "aaa", "aaaa", and "aaaaa". This is not found in a few older instances of regexes. **BRE mode requires \{*m*,*n*\}**. |

#### Regular Expressions Lab

Metacharacters

\w [A-Za-z0-9\_] and often used to match characters in English text.

## Finite State Machines

Mathematical computation model, used to design computer programs and sequential logic

Can only be in one state at a time. Moves between current state and next state with transitions.

Used to represent lexical analyzers

## Lexical Analyzers

Coverts sequence of characters into tokens

Creates a table drive equivalent FSM

Token, represents information that explicitly indicates the categorization.

Lexem – value of token

Category - label of the token type.

**Lex**

Old school

**Flex**

New School

Generates lexical analyzer

**Makefiles**

Order matter, build yacc.o then lex.o then remainined .o files

**Tidbits**

## Parsing

Syntactic analysis

Creates a LALR parser from grammar

Grammars

BNF

Yakk

Bision