PROJECT 1

CHECKPOINT 1

TAs: Alex Bainbridge
Krithika Vijayan

Carnegie Mellon University

AGENDA

- Project 1 Checkpoint 1
- Lex and yacc
- Intro to Git
- Setting up
- Q&A

What are you going to build?

A webserver that can handle multiple concurrent connections!

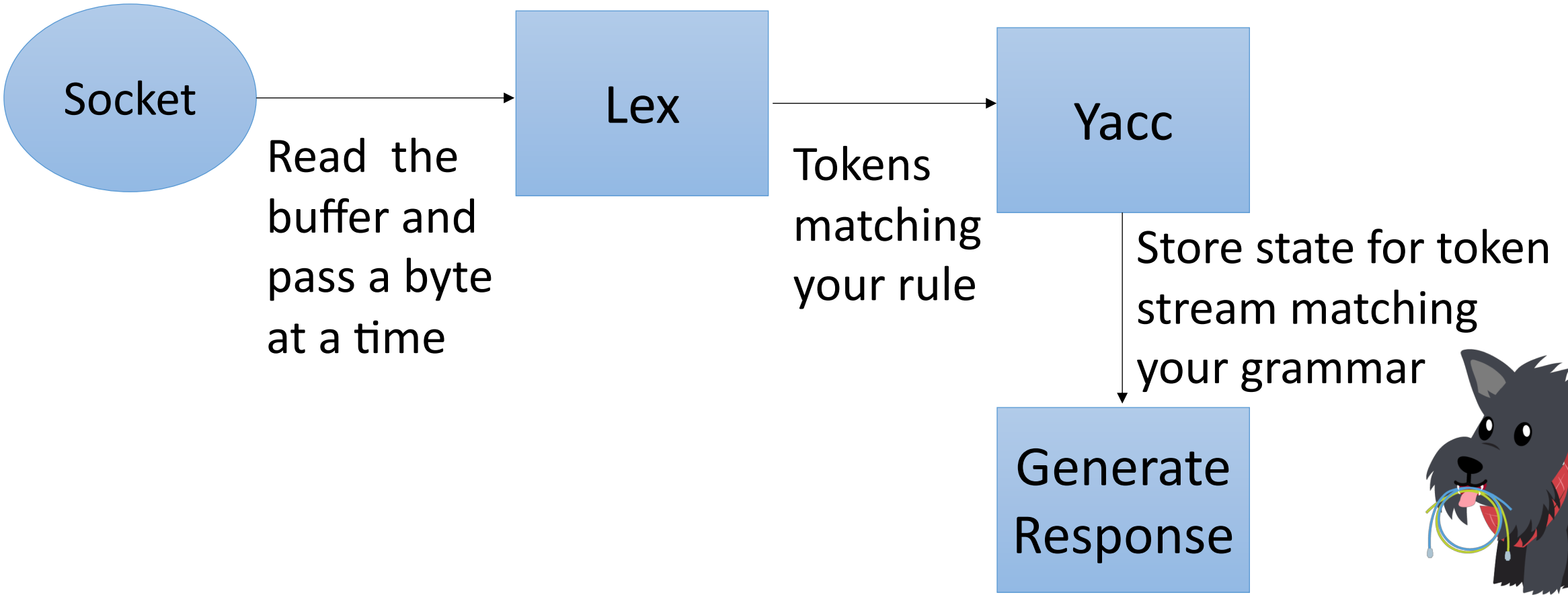
Can you recall building an HTTP 1.0 Proxy for 15-213 ???

Project 1: HTTP déjà vu

However this time….

- It is HTTP 1.1
- select() – based echo server handles multiple clients
- Lex and Yacc for parsing HTTP 1.1 requests

Basic Idea behind Lex and Yacc



Lex

- It's a program that breaks input into sets of "tokens," roughly analogous to words.
- The general format of Lex source is:

  {definitions} ------- Definition of tokens
  %%
  {rules} ---------- for handling the detected token
  %%
  {user subroutines} --- C code(Process tokens)

- The absolute minimum Lex program is thus %% (no definitions, no rules) which translates into a program which copies the input to the output unchanged.

Yacc

- YACC can parse input streams consisting of tokens with certain values.
- YACC has no idea what 'input streams' are, it needs pre-processed tokens.
- A full Yacc specification file looks like:

```
{declarations}  -------      Types of each token
%%
{rules}         ----------   Grammar
%%
{programs}      --------     C code
```

- The smallest legal Yacc specification is

```
%%
rules
```

A VERY SIMPLE EXAMPLE FOR LEX & YACC

- Let's say we have a thermostat that we want to control using a simple language.

```
heat on
        Heater on!
heat off
        Heater off!
target temperature 22
        New temperature set!
```

- The tokens we need to recognize are: heat, on/off (STATE), target, temperature, NUMBER.

Lex tokenizer

```
%{
#include <stdio.h>
#include "y.tab.h"
%}
%%
[0-9]+                  return NUMBER;
heat                    return TOKHEAT;
on|off                  return STATE;
target                  return TOKTARGET;
temperature             return TOKTEMPERATURE;
\n                      /* ignore end of line */;
[ \t]+                  /* ignore whitespace */;
%%
```

YACC GRAMMAR FILE

```
commands: /* empty */
        | commands command
        ;

command:
        heat_switch
        |
        target_set
        ;

heat_switch:
        TOKHEAT STATE
        {
                printf("\tHeat turned on or off\n");
        }
        ;

target_set:
        TOKTARGET TOKTEMPERATURE NUMBER
        {
                printf("\tTemperature set\n");
        }
        ;
```

REMAINING PART OF THE YACC FILE

```
%{
#include <stdio.h>
#include <string.h>

void yyerror(const char *str)
{
        fprintf(stderr,"error: %s\n",str);
}

int yywrap()
{
        return 1;
}

main()
{
        yyparse();
}

%}

%token NUMBER TOKHEAT STATE TOKTARGET TOKTEMPERATURE
```

Intro to Git

Best resource?

Git Cheatsheet provided!

Daily workflow with git

- Check for any remote updates
- Do your work
- Test your work
- Check differences, try to isolate changes
- Commit your work; repeat as needed
- Check for any remote updates
- Push changes, or submit pull request

Translated to git commands

- git pull

Fetching from a remote repository

- vim, emacs, make, create, magic, etc.

Text editors to modify the code

- make test

Run your changes!

- git status

See all changed files

- git diff

Understand differences line by line

- git add

Stage changes, potentially line by line

- git commit -m 'Isolated changes x and y'
- git push

Update the remote repository

Let’s set up together!!

$ ssh andrewid@unix.andrew.cmu.edu

download Project1_starter.tar.gz from the course website and scp it to ~/private

(Ex: scp Downloads/checkpoint1.tar.gz andrewid@unix.andrew.cmu.edu: private/15-441-project-1)

$ tar –zxvf Project1_starter.tar.gz

$ *cd 15-441-project-1*

$ git init

REFERENCES

- https://github.com/theonewolf/15-441-Recitation-Sessions/blob/master/recitation1/recitation1.pdf
- https://github.com/theonewolf/15-441-Recitation-Sessions/blob/master/recitation2/recitation2.pdf
- https://github.com/theonewolf/15-441-Recitation-Sessions/blob/master/recitation3/recitation3.pdf
- http://moss.csc.ncsu.edu/~mueller/codeopt/codeopt00/ y_man.pdf
- https://www.tldp.org/HOWTO/Lex-YACC-HOWTO-4.html