JSH $

JR Aviles & Joe Liccini

Building, Running, and Cleaning

1. Un-tar Source Code:

```
tar xvf jsh.tar.gz jsh/
```

or acquire via git:

```
git clone https://github.com/jeanralphaviles/jsh.git
```

2. Run: `cd jsh/`

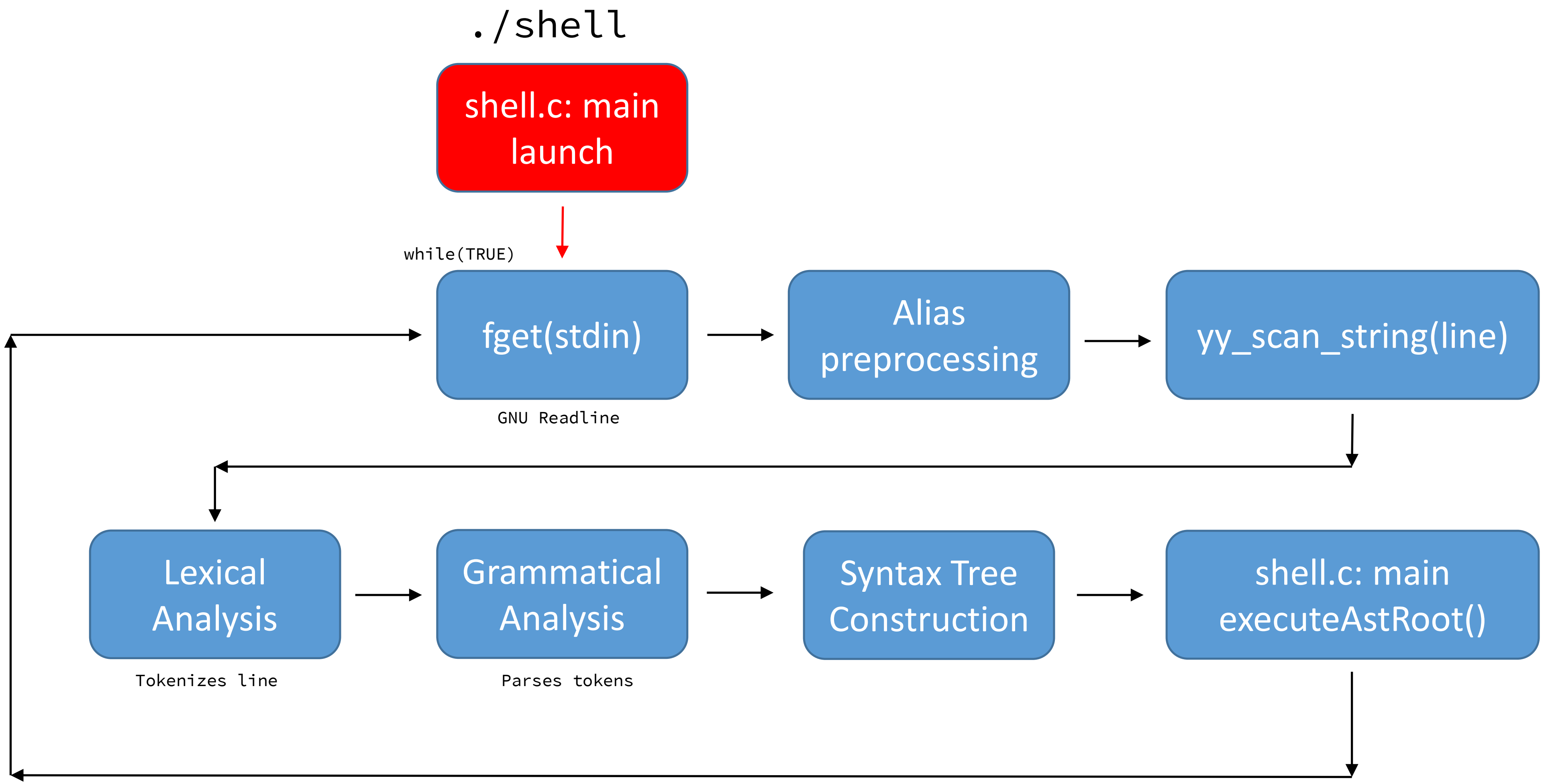
3. Build: `make`

4. Run via: `./shell`

5. Brace yourself

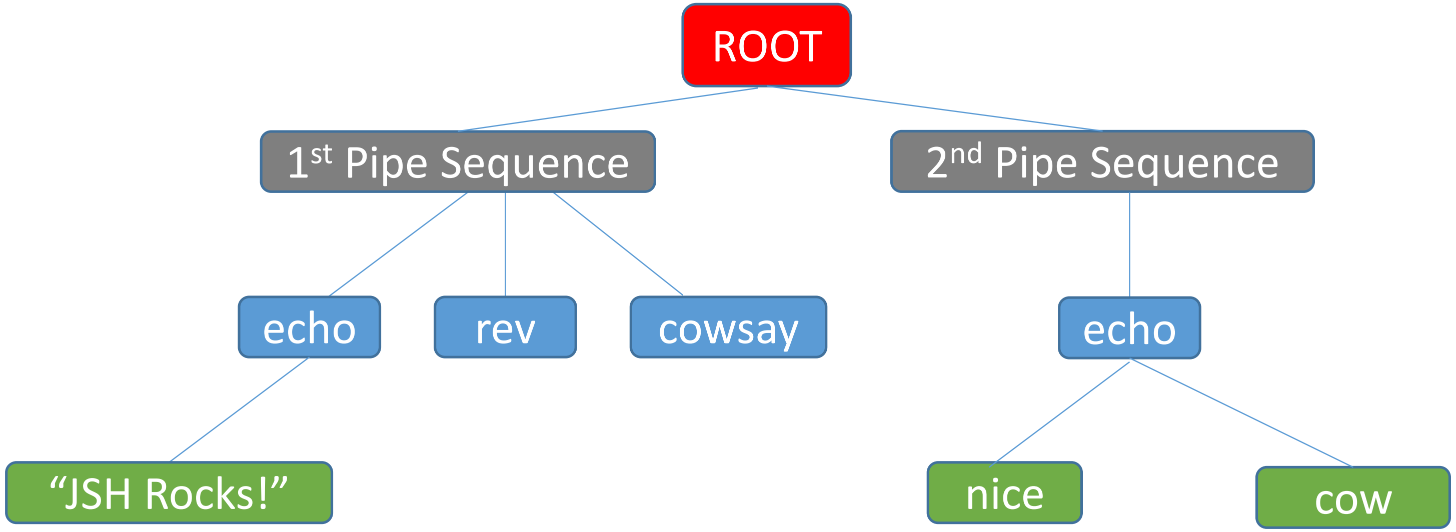
6. Cleanup with: `make clean`

Flow


./shell
shell.c: main launch
while(TRUE)
fget(stdin)
GNU Readline
Alias preprocessing
yy_scan_string(line)
Lexical Analysis
Tokenizes line
Grammatical Analysis
Parses tokens
Syntax Tree Construction
shell.c: main executeAstRoot()

Abstract Syntax Tree Execution

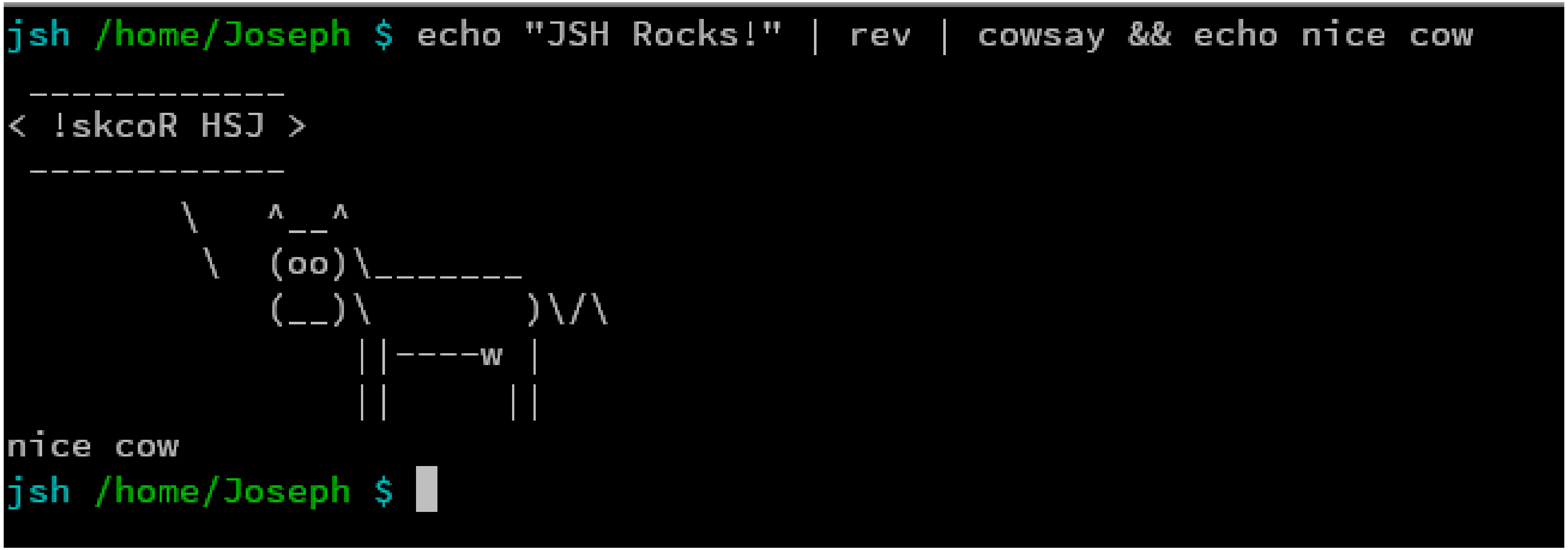
```
jsh $ echo “JSH Rocks!” | rev | cowsay && echo nice cow
```



Key

- Pipe Sequence
- Command
- Argument

Output



```
jsh /home/Joseph $ echo "JSH Rocks!" | rev | cowsay && echo nice cow
 ____________
< !skcoR HSJ >
 ------------
        \   ^__^
         \  (oo)\_______
            (__)\       )\/\
                ||----w |
                ||     ||
nice cow
jsh /home/Joseph $
```

Features

- Quoted String Literals
- Piping
- IO Redirection (via ‘<‘, ‘>’, and ‘>>’)
- `setenv`, `printenv`, `unsetenv`.
- `cd` (no args brings to /home/USER)
- `alias`, `unalias`, (including infinite alias detection)
- `bye`
- stderr redirection and connection to stdout
- Backgrounding via ‘&’
- IO Redirection and piping of builtin commands.
- Environmental Variable Expansion via `${VAR}`

Features, cont.

- Wildcard Matching (* and ?)
- Tilde Expansion
- ‘&&’ and ‘||’ flow control symbols
- Escaping of Characters via prepending ‘\’
- Command Completion using <ESC> key
- Emacs and Vim Key Bindings
- Process Killing via ‘<Control-C>’, ‘<Control-D>’
- Working directory displayed at prompt
- Warnings for builtin commands
- Colored prompt and warnings
- Command History via Arrow Keys