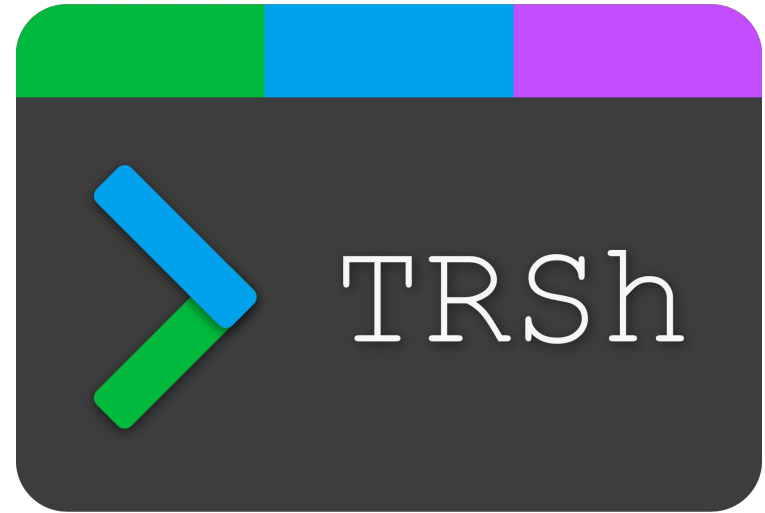


TRShell

OS Projekt FS22

Ruben Hutter, Tobias Hafner



Our project

- > self written linux shell
- > use environment variables to find binaries
- > history of commands
- > built-in utilities
- > autocomplete for binaries, directories and files

Our project

- > self written linux shell
- > use environment variables to find binaries
- > history of commands
- > built-in utilities
- > autocomplete for binaries, directories and files
- python terminal, multi window, splitscreen

Terminal vs. Shell

In the early days: Teletypes (TTY)



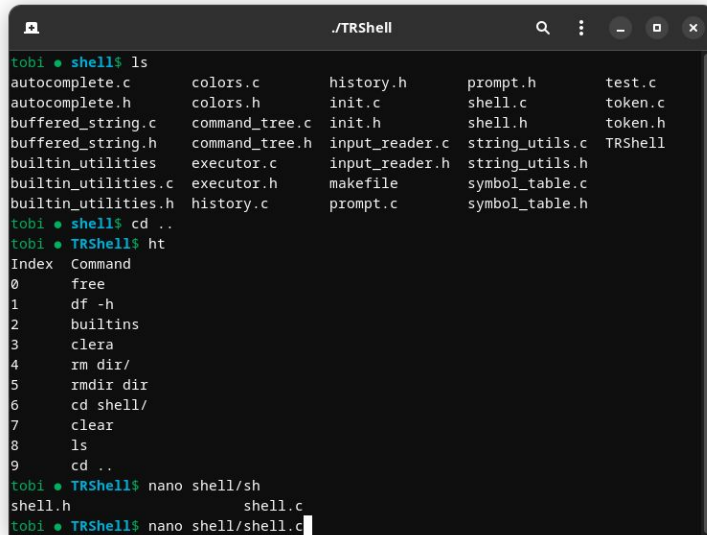
Terminal vs. Shell

Later: Terminals



Terminal vs. Shell

Today: Terminal Emulators (PTY = pseudo teletype)



```
tobi • shell$ ls
autocomplete.c  colors.c  history.h  prompt.h  test.c
autocomplete.h  colors.h  init.c     shell.c   token.c
buffered_string.c  command_tree.c  init.h     shell.h   token.h
buffered_string.h  command_tree.h  input_reader.c  string_utils.c  TRShell
builtin_utilities  executor.c  input_reader.h  string_utils.h
builtin_utilities.c  executor.h  makefile        symbol_table.c
builtin_utilities.h  history.c   prompt.c        symbol_table.h

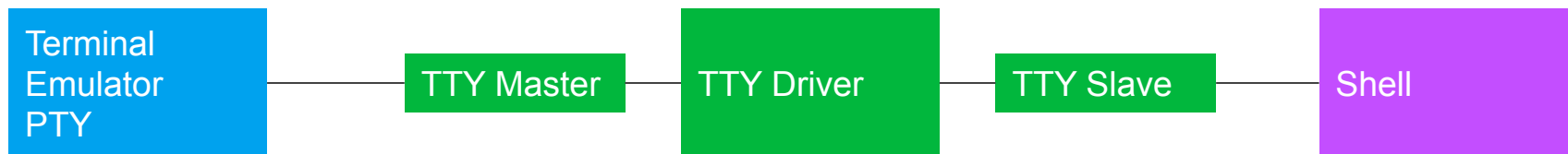
tobi • shell$ cd ..
tobi • TRShell$ ht
Index  Command
0      free
1      df -h
2      builtins
3      clera
4      rm dir/
5      rmdir dir
6      cd shell/
7      clear
8      ls
9      cd ..
tobi • TRShell$ nano shell/sh
shell.h                                shell.c
tobi • TRShell$ nano shell/shell.c
```

Terminal vs. Shell

- > program in user space
- > offers user a way to control the system
- > prompts user for input
- > handles user input
- > returns output of underlying programs or OS

Terminal **and** Shell

— — —



Difficulties

- > screen splitting, multiple instances etc. are terminal feature
- > special control characters to set colors, move cursor, set style of text, ...
- > prototyping of communication using tty driver using python
- > very basic python terminal emulator prototype controlling echo program in C.

Difficulties

— — —

- stopped that path of development
- very time intensive
- would have been whole project on its own

> focused on shell

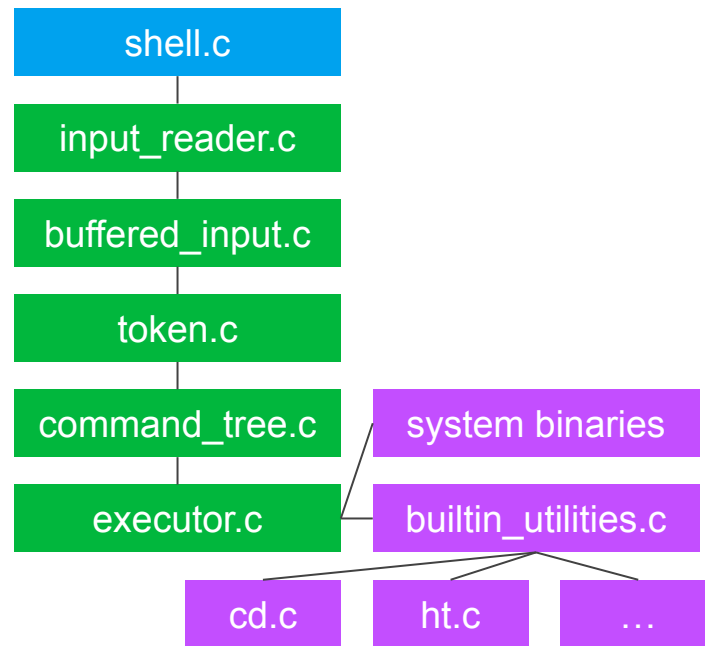
System Overview

- > Command Line Interpreter (CLI)

- > read and process

- > execute

- > built-in-utilities and system binaries



System Overview

> history

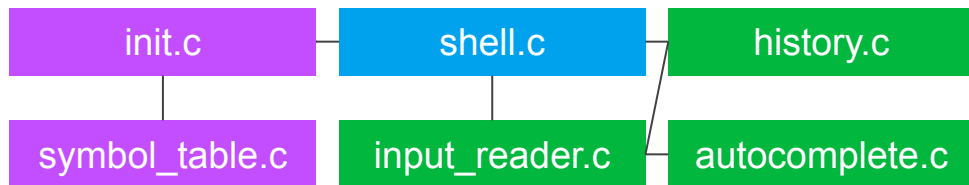
> doubly linked list

> autocomplete

> prompt (\$, #, >)

> symbol table

> hash table



Demo

> loading...

Demo

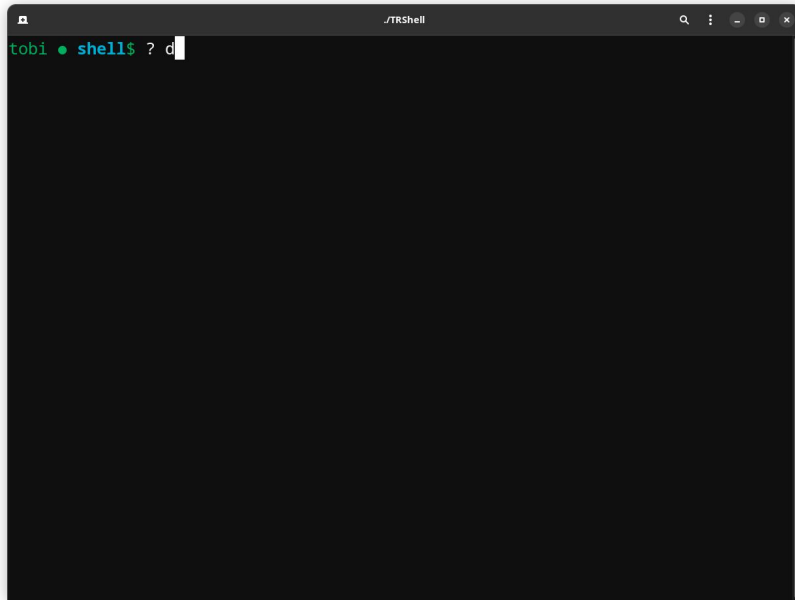
— — —

```
./TRShell
tobi • shell$ ht
Index  Command
0      ls
1      df -h
2      free
3      cd ..
4      cd shell/
5      ht
6      clear
tobi • shell$
```

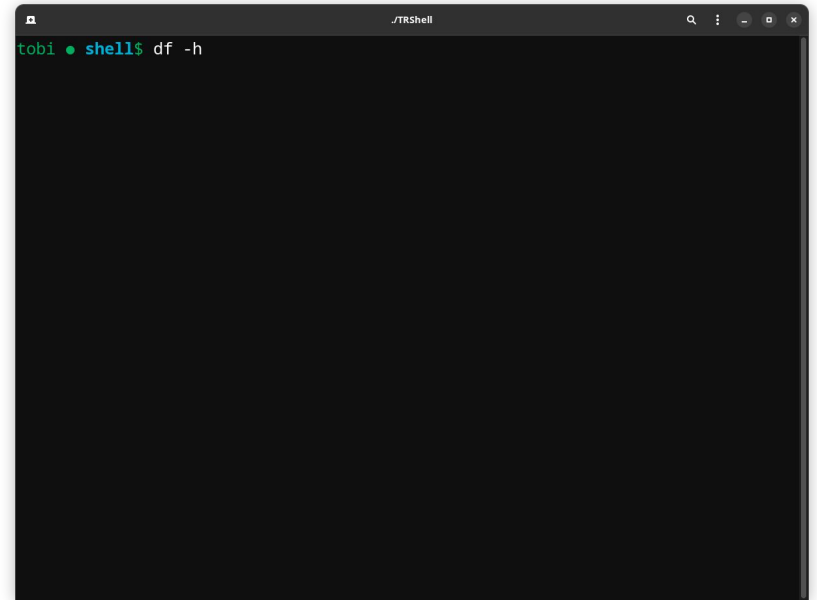
```
./TRShell
tobi • shell$ ht
Index  Command
0      ls
1      df -h
2      free
3      cd ..
4      cd shell/
5      ht
6      clear
tobi • shell$ ht 2
              total      used      free      shared  buff/cache  available
Mem:          3520856    1476972    871828    121312    1172056    1674452
Swap:           0         0         0
tobi • shell$
```

Demo

— — —



A terminal window titled `./TRShell` with a search icon, a menu icon, and window control buttons. The prompt is `tobi • shell$` and the input is `? d`.



A terminal window titled `./TRShell` with a search icon, a menu icon, and window control buttons. The prompt is `tobi • shell$` and the input is `df -h`.

Demo

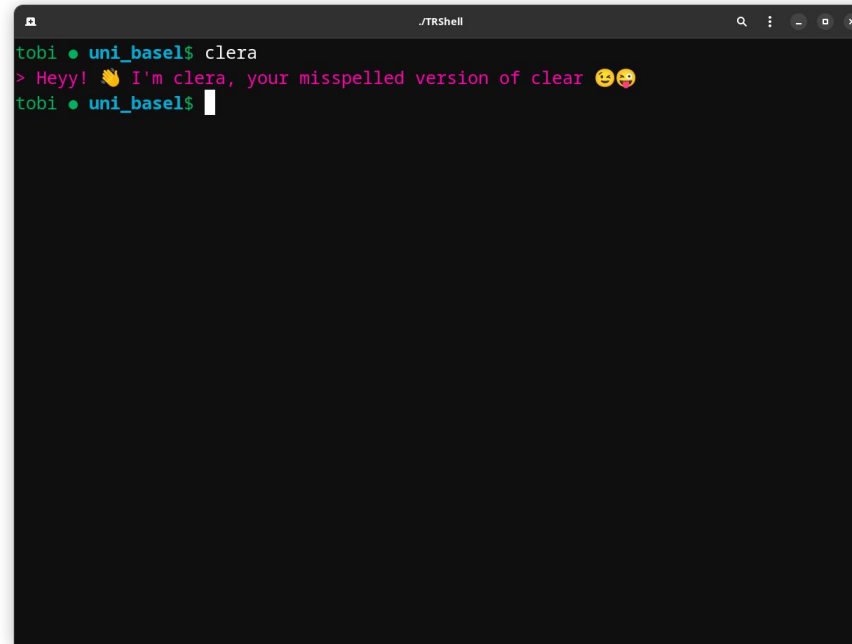
— — —

```
./TRShell
tobi • uni_basel$ cd operating_systems/
TRShell/ OS-Exercises/
tobi • uni_basel$ cd operating_systems/
```

```
./TRShell
tobi • uni_basel$ cd operating_systems/
TRShell/ OS-Exercises/
tobi • uni_basel$ nano operating_systems/TRShell/shell/
autocomplete.h TRShell test.c
command_tree.c autocomplete.c shell.h
string_utils.c prompt.c executor.h
token.h string_utils.h command_tree.h
symbol_table.c buffered_string.c colors.c
symbol_table.h input_reader.c builtin_utilities.c
executor.c token.c history.h
colors.h init.h makefile
history.c buffered_string.h init.c
shell.c prompt.h builtin_utilities.h
input_reader.h builtin_utilities/ .vscode/
tobi • uni_basel$ nano operating_systems/TRShell/shell/
```


Demo

— — —



A screenshot of a terminal window titled ".TRShell". The prompt is "tobi • uni_base1\$". The user enters the command "clera". The output is "> Heyy! 🙌 I'm clera, your misspelled version of clear 😊😊". The prompt is then "tobi • uni_base1\$" with a cursor.

```
tobi • uni_base1$ clera
> Heyy! 🙌 I'm clera, your misspelled version of clear 😊😊
tobi • uni_base1$
```

Lessons learned

- > larger project in C
- > using memory debugger to find mem. leaks, seg. faults...
- > differences between terminal, shell, tty, pty, ...
- > setting realistic expectations
- > project management

Conclusions

— — —

- > interesting expedition into near OS C programming
- > developed appreciation for simplicity and closeness to hardware offered of C
- > satisfied with result

Questions

— — —

```
> man trshell
```