CSCI2467: Systems Programming Concepts

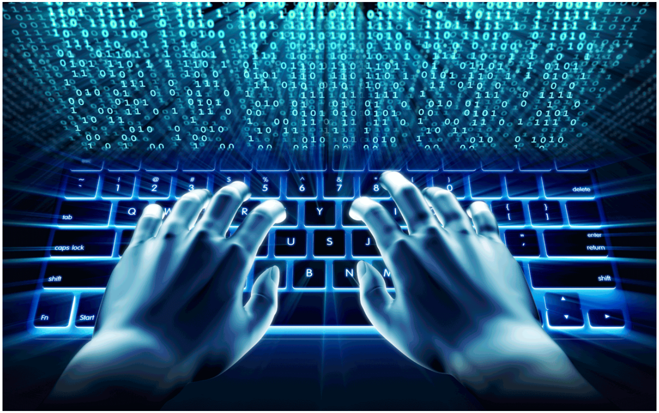
`tsh` checkpoint: next steps

Spring 2020

THE UNIVERSITY *of* NEW ORLEANS

DEPARTMENT OF COMPUTER SCIENCE

First shell lab deadline: can it create processes?



Next steps!

- add support for foreground and background jobs
  - if statement in `eval()` to check value of `bg`
- `builtin_cmd()` and `do_bgfg()`
  - `quit` and `jobs` are easy
  - for `fg` and `bg`: use signals! (next lectures, Section 8.5)
  - note difference between `argv[1]` `argv[1][0]` `argv[1][1]`
- `waitfg()` : see Hints, use while loop and check jobs list using `getjobpid()`
- jobs list! Call `addjob ()` helper function (already given in `tsh.c`)
  - `deletejob()` should get called in `sigchld_handler()`
  - will need to block signals with `sigprocmask` in order to ensure that `deletejob()` is never called before `addjob()`

Next steps: notes

- In `do_bgfg()`: Can't use `switch` on strings (only `ints`)
  Why not? jump table indices
  typical workarounds in C: map strings to ints, or use enum types
- For `tsh`, just use a simple solution for small number of cases:
  ```
  if(!strcmp(s1,s2)) ...
  else if (!strcmp(s1,s2)) ...
  ```
  `s1, s2` can be literal ``‘‘jobs’’`` or variable of type `char *`
  For info on `strcmp` run (from bash): `$ man strcmp`
- Also in `do_bgfg()`: use `atoi()` to convert `char*` to `int`

More notes for later

- don't call `waitpid()` in `eval()` or `waitfg()`
- - call it in `sigchld_handler()`
- - read textbook p.744 about `WUNTRACED` and `WNOHANG`
- `sigchld_handler` uses 3 cases:
- - normal exit `WIFEXITED`
- - ctrl-c (interrupted) `WIFSIGNALED`
- - ctrl-z (stopped) `WIFSTOPPED`

  see text p.745 on using these to check status
- in `do_redirect`: call open/dup2/close
- add a call to `do_redirect()` in correct place in `eval()`

Problems?

- segmentation fault? use `gdb` (see writeup hints)

  - note: `argv[1]` vs `argv[1][1]` vs `&argv[1][1]`

- If you want to use `printf()` for debugging, use:

  `if (verbose)`

- Example from `addjob()`:

```
if (verbose) {
        printf("Added job [%d] %d ... );
}
```

- Now run shell using: `$ ./tsh -v`

Stay focused!

- Actually run your shell as `./tsh` and try doing stuff similar to what is done in tests (trace files), see how it works in `./tshref` also
- Once it *seems* ok to you, then run the tests and see what problems arise
- Not sure what's wrong? See writeup on using `diff`
- also see previous slide, debug using gdb or `printf`

- Once you have most of the commands and handlers implemented, start running more tests and fix bugs

$ `make test01` and `make rtest01`

- `checktsh.pl` will stop on first test that fails
- I will look at source code, I will notice if you pass a test but do it in a way that does not solve the problem
  Also, the usual rules about comments, citing sources and copying code still apply

Finish line

- Due by 23:59 on Thursday March 26
  (to AutoLab, `tsh.c` only)
- 20 traces, 2 points each
- AutoLab will take at least 1 minute to run all traces. Check your results and re-submit.