Quash Tutorial

EECS 678 Staff

Quash in a Nutshell

quash.c

```
main ( … ) {
    while (is_running) {
        CommandHolder *script = parse(&state);

        run_script (script);
    }
}
```

execute.c

```
run_script (CommandHolder* holders) {

    if ( end_condition_reached[1] ) {
        is_running = false;
    }

    for each holder in holders {
        create_process (holder);
    }

    if (holder_contains_forground_jobs) {
        wait_for_all_the_proceses_in_the_job_to_complete;
    } else {
        push_the_job_in_background_job_queue;
    }
}

create_process (CommandHolder holder) {

    Setup_pipes_and_io_redirection_based_on_flags;

    fork_a_child ();

    if (in_child) {
        run_child_command(holder.cmd);
        exit (EXIT_SUCCESS);
    } else {
        run_parent_command(holder.cmd);
    }
}

run_*****_command (Command cmd) {

    switch based on command type {
        case command_type:
            run_command_action (command_arguments);
            break;
        …
        default:
             fprintf (stderr, "Unknown Command\n");
    }
}
```

Essential Data Structures

command.h

```
struct CommandHolder {
     char* redirect_in,
     char* redirect_out,
     int flags,
     Command cmd;
}
```

command.h

```
union Command {
     SimpleCommand simple;
     GenericCommand generic;
     EchoCommand echo;
     ExportCommand export;
     CDCommand cd;
     KillCommand kill;
     PWDCommand pwd;
     JobsCommand jobs;
     ExitCommand exit;
     EOCCommand eoc;
} Command;
```

Example

command.h

```
struct CDCommand {
     CommandType type;
     char* dir;
} CDCommand;
```

Quash Invocation

Example - 1

>> ./quash
[<QUASH_PROMPT>] cd /home/

quash.c

```
main ( … ) {
    while (is_running) {
        CommandHolder *script = parse(&state);

        run_script (script);
    }
}
```

After 3rd line CommandHolder structure array pointed to by script looks like the following:

```
script[0].redirect_in = 0;
script[0].redirect_out = 0;
script[0].flags = 0;
script[0].cmd =>
    script.cmd.type = 5;
    script.cmd.dir = "/home/";
```

Quash Invocation

Example - 2

[<QUASH_PROMPT>] cd /home/ | ls –ll /home/

quash.c

```
main ( … ) {
    while (is_running) {
        CommandHolder *script = parse(&state);

        run_script (script);
    }
}
```

After 3rd line CommandHolder structure array pointed to by script looks like the following:

```
script[0].redirect_in = 0;
script[0].redirect_out = 0;
script[0].flags = 0x10;
script[0].cmd =>
    script.cmd.type = CD;
    script.cmd.dir = "/home/";


script[1].redirect_in = 0;
script[1].redirect_out = 0;
script[1].flags = 0x10;
script[1].cmd =>
    script.cmd.type = GENERIC;
    script.cmd.args = ["ls", "-ll", "/home/"];
```

[<QUASH_PROMPT>] cd /home/ | ls –ll /home/

quash.c

```
main ( … ) {
    while (is_running) {
        CommandHolder *script = parse(&state);

        run_script (script);
    }
}
```

After 3rd line CommandHolder structure array pointed to by script looks like the following:

```
script[0].redirect_in = 0;
script[0].redirect_out = 0;
script[0].flags = 0x10;
script[0].cmd =>
    script.cmd.type = CD;
    script.cmd.dir = "/home/";


script[1].redirect_in = 0;
script[1].redirect_out = 0;
script[1].flags = 0x10;
script[1].cmd =>
    script.cmd.type = GENERIC;
    script.cmd.args = ["ls", "-ll", "/home/"];
```

execute.c

```
run_script (CommandHolder* holders) {

    if ( end_condition_reached[1] ) {
        is_running = false;
    }

    for each holder in holders {
        create_process (holder);
        ---------------------------------
        Iteration-0: Create process for CD
        Iteration-1: Create process for ls
        ---------------------------------
    }

    if (holder_contains_forground_jobs) {
        wait_for_all_the_proceses_in_the_job_to_complete;
        ------------------------------------------------
        NOTE: You need a queue here which is populated
        by create_process () function; to track the
        pids of created processes and wait for them
        to exit
        ------------------------------------------------
    } else {
        push_the_job_in_background_job_queue;
        ------------------------------------------------
        NOTE: Another queue required to accomplish this
        task
        ------------------------------------------------
    }
}

create_process (CommandHolder holder) {

    Setup_pipes_and_io_redirection_based_on_flags;

    pid = fork_a_child ();

    if (pid == 0) {
        run_child_command(holder.cmd);
        exit (EXIT_SUCCESS);
    } else {
        ------------------------------------------------
        NOTE: This is a good place to populate the pid
        queue
        ------------------------------------------------
        run_parent_command(holder.cmd);
    }
}
```

Command Handling

- Parent Side
  - EXPORT
  - CD
  - KILL
- Child Side
  - GENERIC
  - ECHO
  - PWD
  - JOBS

Quash Milestones

- get_current_directory
- create_process (Step-1)
  just uncomment child and parent run process functions
- lookup_env
- run_pwd
- run_cd
- run_export
- run_echo
- run_generic
- create_process (Step-2)
  Setup pipes to establish IO redirection among children
- run_script (Step-1)
  Implement PID-queue, update create_process to track the pids of children. After returning from process creation and if the job is foreground, pop processes one by one from the queue and wait for each of them to exit
- create_process (Step-3)
  Setup file redirection for child process outputs
- run_script (Step-2)
  Implement background job handling
- run_kill
  Implement signal handling in quash to process kill signal

Debugging

- Don’t write a whole bunch of code and then start debugging
- Make progress in small steps!

Add Comments!

If either of these apply to you:

```
// no comments for you
// it was hard to write
// so it should be hard to read
```

```
//When I wrote this, only God and I understood what I was doing
//Now, God only knows
```

Then I won’t hold it against you if you don’t add comments in your code. ☺