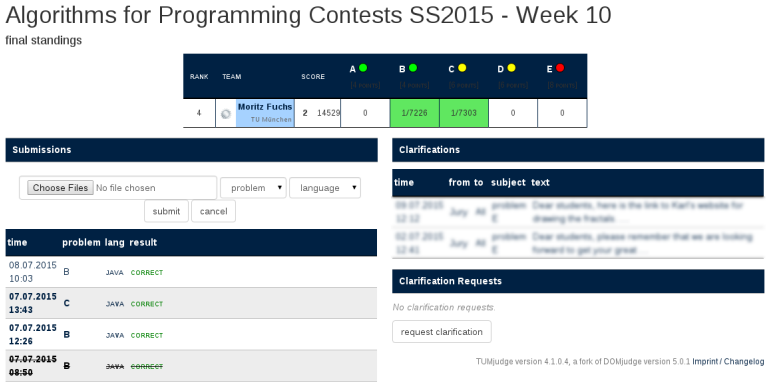
ECE30021/ITP30002 Operating System

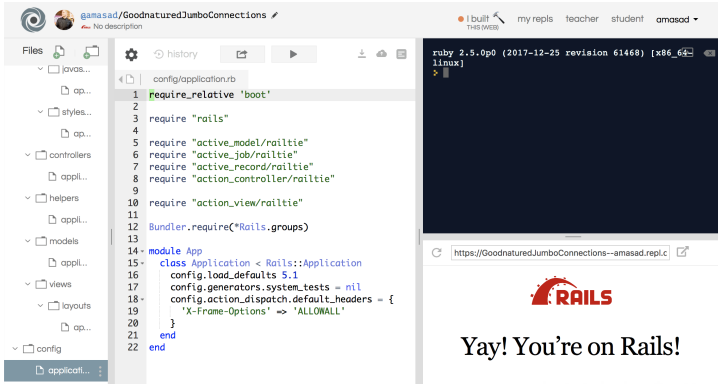
Programming Assignment 2

InstaGrap

InstaGraP: Instantly Grading Programming Assignment

BAEKJOON ONLINE JUDGE





Assignment Overview

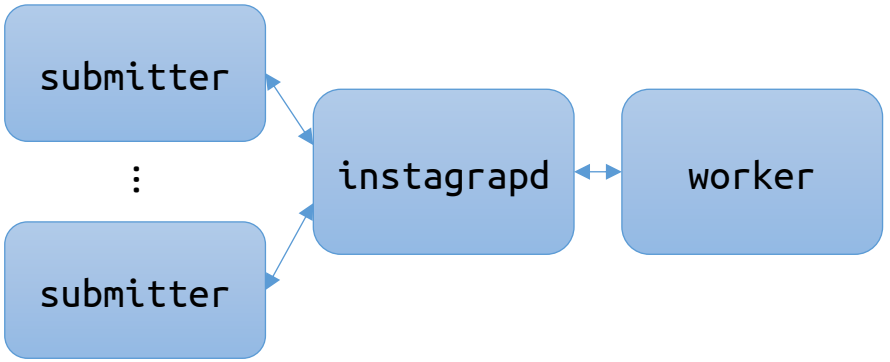
- Construct InstaGraP, a network system that runs tests on a programming assignment submission to give instant feedback
  - a student submits a C program source code file via the client program
  - the server builds and tests the given source code with given test cases
  - the server sends the result back to the student as feedback

- You need to exercise the followings to accomplish this assignment
  - process control (e.g., fork)
  - signal handling
  - inter-process communication using pipe
  - socket programming
  - multithreaded programming

- PA2 will be done as a teamwork of 2 persons

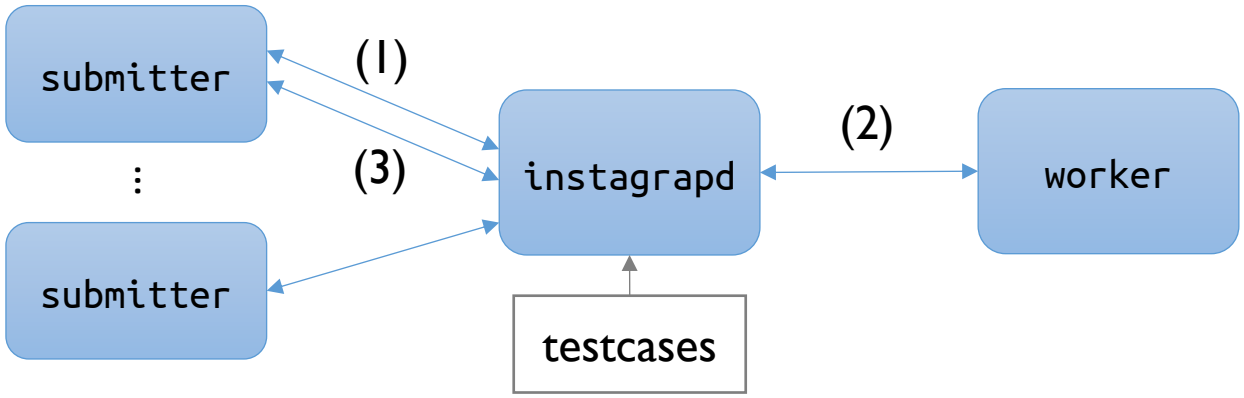
InstaGrap: System Structure

- Components
  - master: `instagrapd`
  - sandboxed worker: `worker`
  - client: `submitter`



- Users
  - admin
    - admin launches a `worker` process
    - admin launches a `instagrapd` process with test cases
    - there is only one admin in the whole system
  - students
    - a student runs `submitter` to turn a C source code file in `instagrapd`
    - a student gets result messages from the server via `submitter`
    - there may be multiple students each of which has a unique student ID
    - multiple students can make submissions simultaneously

InstaGrap: Workflow



1. `submitter` sends a request to `instagrapd` for evaluating a C file with student info and a target program source code file
2. `instagrapd` delivers the C file with a test input to `worker`. Then, `worker` builds and runs the C file with the given input, and sends back the output
    - repeat this step for each test case
3. `submitter` asks `instagrapd` whether the result is ready. If it's ready, `instagrapd` sends a feedback (which is based on the results from `worker`) back to `submitter`

InstaGrap: worker

- Command-line Interface

```
./worker -p <Port>
```

  - `<Port>` port for listening (e.g., 8080)

- Behaviors
  - `worker` receives a pair of a target C file and a test input file
  - Then, `worker` builds a target C file. If fails, it should send the build failure message back to `instagrapd`
  - Once build succeeds, `worker` runs the target program with the given input, and then sends the output back to `instagrapd`
    - A target program always receives input from the standard input, and prints out the result to the standard output
  - If the target program execution takes more than 3 seconds, `worker` stops the execution and returns timeout message to `instagrapd`
  - `worker` should clean up the target program source and binary files after each test (for sandboxing)

InstaGrap: submitter

- Command-line Interface

```
./submitter -n <IP>:<Port> -u <ID> -k <PW> <File>
```

  - `<IP>` IP address of `instagrapd` (e.g., 127.0.0.1)
  - `<Port>` port number of `instagrapd` (e.g., 8090)
  - `<ID>` student ID as a 8 digit number (e.g., 21700999)
  - `<PW>` password as a 8 digit alphanumeric string (e.g., abcd1234)
  - `<File>` a target C source code file

- Behaviors
  - `submitter` connects to `instagrapd` in TCP
  - At first connection, `submitter` transmits a request for evaluating a target source code file to `instagrapd`
  - After that, `submitter` frequently connects with `instagrapd` to receive the feedback from `instagrapd`. Once received, displays it on standard output.

InstaGrap: instagrapd

- Command-line Interface

```
./instagrapd -p <Port> -w <IP>:<WPort> <Dir>
```

  - `<Port>` port for listening of `instagrapd` (e.g., 8090)
  - `<IP>` IP address of `worker` (e.g., 127.0.0.1)
  - `<WPort>` port of `worker` (e.g., 8080)
  - `<Dir>` a path to a testcase directory

- Testcase
  - A testcase directory always contains 20 files whose names are `1.in`, `1.out`, `2.in`, `2.out`, ..., `10.in`, and `10.out`
  - *n*`.in` is a test input (would be given to standard input) and *n*`.out` is the expected output (from standard output)

- Behaviors
  - `instagrapd` listens at a port to one or multiple `submitter`
  - `instagrapd` requests `worker` to run the target program with a test input at a time
  - `instagrapd` rejects a request if it gives a wrong password for a student (different from the one that is given at the submission)
  - Once all testing is done, `instagrapd` answers to `submitter` by giving the number of test cases that the target program passes; or, sends back the build failure message

InstaGrap: Other Requirements

- A target C program reads input from the standard input and writes output to the standard output
- `instagrapd` should be able to communicate with multiple `submitter` instances concurrently
- `instagrapd` accepts any request from `submitter` with a new student ID and a new password
- For invalid inputs, each component should responses with proper error messages
- Communication between processes must be implemented either with pipe or with socket
  - not using File

Assignment

1. Construct InstaGrap by implementing the three modules `instagrapd`, `worker` and `submitter`
   - give `Makefile` together

2. Show that your implementations fulfill all posed requirements by taking video demo
   - devise demo scenarios to cover various cases

3. Describe your design and implementation in write-up
   - Describe your protocols of network communication
   - Support that you used proper techniques to address different requirements

4. Discuss problems/issues/limitations of InstaGrap as an automated grading system in write-up
   - e.g., indicate a problem, suggest a change, propose a new feature

Notes

- `worker` is intended to run in a different account/machine than `instagrapd` for a perfect isolation
  - block any chance of a target program to access a testcase file
- Recommend to use `peace.handong.edu`
  - TA will use peace to run your submitted program for evaluation
  - Ubuntu 16.04.6 kernel 4.15.0
- You cannot connect to peace at an arbitrary port outside of the school network due to firewall
  - still, inside peace, a program can access to any port
- The related example programs are found under `sysprog` in the course Git repository
  - see `IPC/` and `Pthread/`
- Use a port in 8000—9999 for `instrgrapd`

Team

- Basically, two persons will be assigned to one team
  - your partner will be randomly assigned, while excluding the PA1 partner

- Request of consideration: by 11:59PM, 5 April (Fri)
  - send an email to hongshin@handong.edu
  - if you want to avoid pairing with a certain colleague for a personal issue, or
  - if you are willing to work alone (as a one-person team) for a personal issue

- Tentative PA2 partners will be announced at 6 April (Sat)
  - you should contact with the partner and kickoff the work by 8 April (Mon)
  - you can reclaim the team if you cannot contact with the partner by 8 April
  - the teams will be finalized by 9 April (Tue)

Schedules

- Fri 5 Apr
  First announcement
- Fri 5 Apr, 11:59 PM
  Request of consideration
- Sat 6 Apr
  Tentative team partner announcement
  - kickoff a team work by 8 Apr (Mon)
- Tue 9 Apr
  Team finalization
- Tue 9 Mar—Fri 24 Mar
  TA help session by appointment
- Fri 26 Apr, 11:59 PM
  Submission deadline
  - late submission is accepted only within the next 24 hours with 30% penalty

Submission

- Your submission must include the followings
  - write-up: up to 6 pages (either in single- or double-columns)
    - your write-up will be open for peer evaluation
  - URL of your video demo (e.g., YouTube)
    - put the URL in your write-up
  - all related source code files

- How to submit
  - upload your files to a homework repository in Hisnet
  - by only one of the team member

Evaluation

- Points
  - Fulfillment of requirements 30%
    - Check whether you use a proper technique to address each requirement
  - Clarity in technical description 20%
  - Novelty in discussion 20%
  - Soundness of demonstration 20%
  - Peer evaluation (voting) 10%
  - Best peer review award up to extra 10%

- Notes
  - Evaluation will be primary based on your write-up and video demo
  - TAs will test the submitted files on the peace server

Useful Links

- Linux man pages
  https://linux.die.net/man/

- GNU C library reference
  https://www.gnu.org/software/libc/manual/pdf/libc.pdf

- Example code from *The Linux Programming Interface*
  http://man7.org/tlpi/code/online/all_files_by_chapter.html

- Socket programming
  https://www.cs.dartmouth.edu/~campbell/cs60/socketprogramming.html

Little Help from TA

- TA's
  - Mr. Jeewoong Kim jeewoong@handong.edu
  - Ms. Juyoung Jeon 21931009@handong.edu
- Services
  - Help you use the Peace server
  - Explain the related example programs
- How to contact
  - ask a question on Piazza
  - make an offline meeting appointment via Piazza (as a public post)
    - less than 30 minutes, open to every one