Nachos project

Vincent Danjean and Vania Marangozova

décembre 2017

Goal of this presentation

- Introduction to the project
  - quick description of the 5 steps
  - quick description of the provided software
- Explanation about the organization during the project
- Open questions
- Lots of slides :
  - not all of them will be detailed
  - read them (and re-read them later) when you have some difficulties

Topic

Groups definition

- 4 students per group with one leader
- mail the contact address (see later) today with the 4 names and the leader
- leader role :
  - intermediate between the team and the supervisors
  - the leader must supervise all team activity
    - if one member is missing, the leader must know it (and report)
  - the leader takes the decision when there is arguments
- in case of problems within a team, the supervisors must be advised as soon as possible

Preliminary required steps

Moodle registration

- register to the SYSPROJ course
  https://im2ag-moodle.e.ujf-grenoble.fr/course/view.php?id=126
- set your cursus : M1 Mosig or M1 Info
- set your team, according to what has just been decided

=> must be done at the beginning of this afternoon

Git repository

- find somewhere to host a git repository (github, gitlab, etc.)
- put your Git URL for us in Moodle
  If the Git is private :
  - if the server is private, we will need an account
  - else, we will give you our login

=> must be done before tomorrow morning, else ask for a meeting

Topic

Goals

- To understand the internal behavior of operating systems
- To handle a large software
- To work within a team : project management
- To program main OS mechanisms

Environments

Working environment

- Nachos : pedagogic system
- work in a simple virtual machine
- can be debugged with usual tools (gdb, valgrind, etc.)
- deterministic, easy to modify, easy to test

Provided environment

- mandelbrot.e.ujf-grenoble.fr : all is working
- an install script for Debian/Ubuntu with "as-is" (no guaranty)

=> Work on Mandelbrot in case of problems

Step 1 : First Steps

- Installation
- First Compilation
- First observation

Tips

- come back again to this steps when you will need to debug more

Step 2 : Input/Output

- Nachos has a I/O hardware (like a serial line)
- Goal of this step :
  - play with the hardware (part 2)
  - write a (kernel) driver for this hardware (part 3)
  - write I/O system calls and user programs (part 4)
  - extend the system calls (string handling, etc.) and understand some mechanism (Nachos process halting, etc) (next parts)

Tips

- the part 1 is what is expected to work after the part 4, do not try to do it immediately

Step 3 : Multi-threading

- Nachos is able to run several kernel threads (with no user code)
- You must implement a threading model (user threads that will be mapped 1-to-1 to kernel threads, as in most usual OS)

Tips

- read the following step to avoid to do bad design choices

Step 4 : Virtual Memory

- set up (simplified) pagination
- start several (multi-threaded) processes

Tips

- at the end of this step, you should be able to run a mini-shell

Steps 5 and 6

Step 5 : Filesystem

- the FS provided by Nachos is limited :
  - one directory
  - 10 files maximum
  - only small files
- do better

Step 6 : Network

- Nachos can communicate with a non-reliable network
- Implement a reliable way to communicate (as TCP but less complex)

Nachos

Initial source code provided as a Git repo

- you must clone our Git repository
  So, we can easily ask you to pull to get hot-fixes
  => request a meeting tomorrow if this is difficult for you
- you must regularly (each day) publish your work in the Git repo you referenced on Moodle

Global structure

Sources compile :

- a linux program with the NachOS OS and the virtual machine
- some MIPS programs that will be loaded and executed as user space programs by the virtual machine

=> the (NachOS) OS is plain C(++) code executed natively on Linux.
Use gdb, etc. to debug it
=> user code is executed through the VM (see step 1 on how to debug it)

Code organization

directories in |code/|

- [`bin`] utilities used during the compilation. Should not be modified
- [`machine`] virtual machine. Should not be modified
- [`network`, `userprog`, `filesys`, `threads`, `vm`] the NachOS OS, can/must be modified
- [`test`] user programs. They will be (cross-)compiled as MIPS executables. You will add your own test programs
- [`build`] directory where all object and program files are put

flavors

- NachOS can be compiled with different features
- Initially, these features are conflicting together, hence several flavor of NachOS are compiled
- Ideally, your final NachOS will be able to have all features
- Read `code/README.Makefiles`

Topic

Time Schedule

Suggested planning

- Step 1 : two days (including this one)
- Step 2 : two days
- Step 3 and 4 : four days each
- Step 5 and 6 : four days each, but in parallel
- Final developments and tests, defense preparation : four days

Tips

- at the end of the project, step 2 can be done in 1 hour...
- you will miss time. Go ahead when you can
- do not starve on any point. Communicate (see later) with us

Supervisor contact

First by mail

- either the Moodle forum (support)
- or `cse-nachos@lists.forge.imag.fr`
- but never personal email, even when answering
  - we are not available at the same time, we must follow all what happens
- always put a subject starting with [NachOS/Team X]

Physical meeting in room "Projet System" on ADE

- rooms "Projet libre service" on ADE can be used (without us in them)
- we must see each team at least once per week
  - it is up to you to respect this rule
- we are not always available for physical meeting
  - a meeting should be requested at least 24h before with a mail describing the issue
  - work on other parts when you are really blocked by an issue

Reporting difficulties, asking for help

Report any problems

- missing software on mandelbrot, missing student in the team, conflicts in the team, etc.
- we probably won't solve all problems, but we surely won't solve problem we don't know about

In case of a blocking bug, write a clear bug report

All the following information must be present if you want some help

- a git commit (or tag/branch name) on your git where the bug occurs
- a description of what to do to reproduce the bug
- a explanation of what you expect
- a explanation of what you observe
- a description of what you have already done/analyzed

The clearer your message is, the quicker you will receive help

Lots of students (and teams)

- this year, there are lots of students for this project
- you must be pro-active in communication
- do not wait to much before requesting help
  - but do not forget to signal if you find the solution

Some days with only mail support

- in particular, during the second week, you will have help mainly from mail (both supervisors with other duties except on Monday and Friday)
- possible (limited) mail support from January, 3rd (even if holidays end on 7th)

Coding

- few code to provide, but a lot to think
- the goal is not that something nearly work
- the goal is that what works correctly works and that you know how and why
- the design of your proposals is crucial
- the documentation of your design will be an important part of your evaluation

The evaluation is described on Moodle (near the end)
Before Christmas, you will have to provide the start of your documentation/report.

Code management

- use Git
  - ask for help for any Git issue
  - once commited, any work can generally be restored, even if someone makes some mistakes with branch management. Just ask.
- regularly publish on your Git repo
  - this is also a kind of backup
- look at the planning. Avoid to be too late
- do early integration
- do early tests
- do early integration and tests
- do some tests

tips

- debug with `gdb` and not `printf`
  Even if you have to learn `gdb`, you will gain some time
- did I tell you to do some tests ?

Work within each step

- do required parts of all steps
- choose some additional parts and/or choose you own extension
  - in the latter case, tell us before to be sure it can be done
- you will be evaluated on your idea and design more than on you code

Note

Before the use of Git, students were asked to mark their modifications with `#ifdef CHANGED`. It is not done anymore (Git is way better), you can ignore any reference to it if you find some.

Topic

Plagiarism/copy

- you need to write your code
  - no problem to discuss about issues/solution with other people (us, other teams, programming forums, etc.)
  - no problem to read other code
  - problem if external code is copied into your git
- We will have to evaluate your work
  - we know some NachOS git repo can be found
  - if we find plagiarism, we immediately delegate the issue to the "section disciplinaire" of UGA
  - it already happened :-(

Conclusion

- time is short
  this project is designed for 4 weeks of 4 full-time students

=> you should be ready to start

- enjoy this project

Questions ?