SCALE FOR PROJECT PROCESS AND MEMORY (/PROJECTS/PROCESS-AND-MEMORY)

Introduction

To ensure this evaluation goes smoothly, please respect the following set of rules :

- Please remain courteous, polite, respectful and constructive at all times during this exchange. The trust bond between the school's communauty and yourself depends on it.

- Should you notice any malfunctions within the submitted project, make sure you take the time to discuss those with the student (or group of students) being graded.

- Keep in mind that some subjects can be interpreted differently. If you come accross a situation where the student you're grading has interpreted the subject differently than you, try and judge fairly whether their interpretation is acceptable or not, and grade them accordingly. Our peer-evaluation system can only work if you both take it seriously.

Guidelines

- You may only evaluate whatever is in the GiT submission directory of the student you are grading.

- Make sure to check wether the GiT submission directory belongs to the student (or group) you're grading, and that it's the right project.

- Make sure no mischievous aliases have been used to trick you into correcting something that is not actually in the official submitted directory.

- Any script created to make this evaluation session easier - whether it was produced by you or the student being graded - must be checked rigorously in order to avoid bad surprises.

- If the student who is grading this project hasn't done the project him/herself yet, he/she must read the whole topic before starting the evaluation session.

- Use the flags available to you on this scale in order to report a submission directory that is empty, non-functional, that contains a norm errors or a case of cheating, etc...
In this case, the evaluation session ends and the final grade is 0 (or -42, in case of cheating). However, unless the student has cheated, we advise you to go through the project together in order for the two (or more) of you to identify the problems that may have led for this project to fail, and avoid repeating those mistakes for future projects.

- Careful ! The code testing is done in the student hand made linux distribution.

Attachments

Subject (https://cdn.intra.42.fr/pdf/pdf/3980/lk_process_and_mem.en.pdf)

Mandatory

Free points

Check the student hand made linux distribution. Is the kernel a linux one ?

Check the version too, must be 4.x

Yes No

Installation

Is there a Makefile to copy the syscall tables, the source of the syscall ?

Does it work ? Does it compile the kernel and boot on it ?

☑ Yes ☒ No

Syscall

You are a real developer now

Base

Compile the example given by the student (Verify if it is not reading from /proc/ instead of using the syscall.)
Execute it, and check if the following information is here:
- PID
- State of the process (Something like PID_UNRUNNABLE, PID_RUNNABLE, PID_STOPPED)
- Age of the process, user friendly or not (timestamps, seconds, ns, tick are okay)
- Pointer on the stack of the process. (The display must be the adress of the stack)
- Try to dereference the stack pointer. Should SEGV.
- An array of ints fill with childs PID
- Pid of the father
- Root path of the process
- Full path of the current pwd of the process

Rate it from 0 (failed) through 5 (excellent)

it++

Does the example iterate over the children of pid ?
And the parents ? (To the kernel, pid 0)

☑ Yes ☒ No

Bonus

Bonus functions

Does the student have implemented another syscall ? Like fork, wait

Rate it from 0 (failed) through 5 (excellent)

Ratings

Don't forget to check the flag corresponding to the defense

✔ Ok

★ Outstanding project

Empty work

Incomplete work

No author file

Invalid compilation

Norme

Cheat

Crash

Forbidden function

Conclusion

Leave a comment on this evaluation

Preview!!!