Remember that the quality of the defenses, hence the quality of the of the school on the labor market depends on you. The remote defences during the Covid crisis allows more flexibility so you can progress into your curriculum, but also brings more risks of cheat, injustice, laziness, that will harm everyone's skills development. We do count on your maturity and wisdom during these remote defenses for the benefits of the entire community.

SCALE FOR PROJECT NM-OTOOL (/PROJECTS/42CURSUS-NM-OTOOL)

You should evaluate 1 student in this team

★

Git repository

```
git@vogsphere.msk.21-school.ru:vogsphere/intra-uuid-164d5
```

Introduction

Please follow the following rules :

- Remain polite, courteous, respectful and constructive throughout the evaluation process. The well-being of the community depends on it.

- Identify with the person (or the group) evaluated the eventual dysfunctions of the work. Take the time to discuss and debate the problems you have identified.

- You must consider that there might be some difference in how your peers might have understood the project's instructions and the scope of its functionalities. Always keep an open mind and grade him/her as honestly as possible. The pedagogy is valid only and only if peer-evaluation is conducted seriously.

Guidelines

- Only grade the work that is in the student or group's GiT repository.

- Double-check that the GiT repository belongs to the student or the group. Ensure that the work is for the relevant project and also check that "git clone" is used in an empty folder.

- Check carefully that no malicious aliases was used to fool you

and make you evaluate something other than the content of the official repository.

- To avoid any surprises, carefully check that both the evaluating and the evaluated students have reviewed the possible scripts used to facilitate the grading.

- If the evaluating student has not completed that particular project yet, it is mandatory for this student to read the entire subject prior to starting the defence.

- Use the flags available on this scale to signal an empty repository, non-functioning program, a norm error, cheating etc. In these cases, the grading is over and the final grade is 0 (or -42 in case of cheating). However, with the exception of cheating, you are encouraged to continue to discuss your work (even if you have not finished it) in order to identify any issues that may have caused this failure and avoid repeating the same mistake in the future.

- Remember that for the duration of the defence, no segfault, no other unexpected, premature, uncontrolled or unexpected termination of the program, else the final grade is 0. Use the appropriate flag.
You should never have to edit any file except the configuration file if it exists.
If you want to edit a file, take the time to explicit the reasons with the evaluated student and make sure both of you are okay with this.

- You must also verify the absence of memory leaks. Any memory allocated on the heap must be properly freed before the end of execution.
You are allowed to use any of the different tools available on the computer, such as leaks, valgrind, or e_fence. In case of memory leaks, tick the appropriate flag.

Attachments

subject.pdf (https://cdn.intra.42.fr/pdf/pdf/13307/en.subject.pdf)

Preliminaries

You will use the following code to test and you will compiled the second one in 32bit format $> cat test_facile.c #include <stdio.h> int main(void) { puts("Test facile"); return (0); } $> cat test_moins_facile.c #include <stdio.h> int une_globale = 40; int main(void) { printf("La globale vaut: %d\n", une_globale); return (0); } $> cc test_facile.c -o test_facile $> cc test_moins_facile.c -o test_moins_facile $> cc -m32 test_moins_facile.c -o test_moins_facile_32-bit $> In case you dont know, to identify the type and architecture of a file: man 1 file If you ve got issue to find a universal binary, a search in the PATH: (IFS=$'\n'; for d in ${PATH//:/$IFS}; do find "$d" -type f -exec file '{}' \+ | grep -i -A3 universal ; done) To create a universal binary: clang/gcc: -m32 pour cross-compile a 32-bit lipo -create -output <universel> <binaire arch. 1> <binaire arch. 2> ... If you cant find a dynamic librairy (.so, .dylib): find /usr/lib -type f -iname '\.dylib' 2>/dev/null*

Preliminary tests

Please check the following first:

- Are the files in the right place in the git repo
- autor file is valid
- Le Makefile est présent et compile bien les exécutables
- The Makefile work as intended and compile the right binaries
ft_nm et ft_otool
- No cheat (no forbidden functions, the student can explain his code...)

If something doesnt follow the subject, the mark for the evaluation will stop there.
But you can keep evaluating and you are strongly suggested to talk about the project
a bit further

☑ Yes ☒ No

---

nm tests

Nm easy test

Test ft_nm with the binary test facile. The output should be
in accordance with the true nm.

☑ Yes ☒ No

Test less easy

Test ft_nm on the "test moins facile" binaries. On the 32 and 64 bits binaries
the ft_nm output should be the same as nm (no output when using diff)

☑ Yes ☒ No

Other Test

ft_nm output is always equal to the real nm output, with any test.
(no output when using diff)

☑ Yes ☒ No

Multiples arguments

ft_nm can take multiple arguments

☑ Yes ☒ No

Object files

Test ft_nm with 32 and 64 bits object files (.o).
Order can differ with the real nm.

Yes No

Dynamic librairy

Test ft_nm with a dynamic librairy (*.dylib *.so)
(Look in /usr/lib/). The output should look like the real nm output but
the symbol order can be arbitrary

Yes No

Universal binary

Test ft_nm on a universal binary (example::
/usr/bin/python). Dont forget to use the command "file ".
The output should be the same as the real nm. But the order of the symbols can differ.

Yes No

otools

Otool easy test

test ft_otool with the "test facile" binary. the output should be
like otool -t
(no output when using diff)

Yes No

Other

On 32 and 64 bits binaries, the ft_otool output should be the same
as otool -t
(no output when using diff)

Yes No

multiples arguments

ft_otool should be able to take multiple arguments

☑ Yes ☒ No

object file

Test ft_otool with some 32 and 64 bits object files (.o).
output should be like otool -t

☑ Yes ☒ No

Dynamic librairy

Test ft_otool on dynamic librairies (*.dylib *.so)
(look in /usr/lib/). Output should be like the otool one
-t.

☑ Yes ☒ No

Universal binary

Test ft_otool with universal binary (example:
/usr/bin/python). Dont forget to use the command "file ".
The Output should be like the otool one

☑ Yes ☒ No

Bonus

Des bonus

Count one point per bonus done and
Maximum 5 bonus.

Each bonus must be:

- useful (at your discretion)
- work flawlessly

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 | No author file | Invalid compilation | Norme | Cheat | Crash

⊘ Forbidden function

Conclusion

Leave a comment on this evaluation

Finish evaluation