(https://profile.intra.42.fr)

SCALE FOR PROJECT WOODY WOODPACKER (/PROJECTS/WOODY-WOODPACKER)

Introduction

Please respect the following rules:

- Remain polite, courteous, respectful and constructive throughout the correction process. The well-being of the community depends on it.
- Identify with the person (or the group) graded 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 correcting and the corrected students have reviewed the possible scripts used to facilitate the grading.

- If the correcting student has not compl			
project yet, it is mandatory for this stude			
entire subject prior to starting the defend	e.		
- Use the flags available on this scale to	signal an empty repository,		
non-functioning program, a norm error,			
the grading is over and the final grade i	s 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 sam	e mistake in the future.		
- The evaluation must take place in a vir	tual machine under Linux with a		
kernel > 3.14 and admin (root) rights.			
Attachments			
_			
Subject (https://cdn.intra.42.fr/po	lf/pdf/5670/woody_woodpacke	r.en.pdf)	
resources.tar (/uploads/documen	/document/339/ressources.tar)		
	/document/339/ressources.tar)		
Sources	/document/339/ressources.tar)		
Sources Preliminaries Preliminaries	/document/339/ressources.tar)		
Sources Preliminaries	/document/339/ressources.tar)		
Sources Preliminaries Preliminaries First, check that the following is true:			
Sources Preliminaries Preliminaries			
Sources Preliminaries Preliminaries First, check that the following is true:	GiT repository.		
Sources Preliminaries Preliminaries First, check that the following is true: - Something has been uploaded to the	GiT repository. n ASM).		
Sources Preliminaries Preliminaries First, check that the following is true: - Something has been uploaded to the orange of the project is written in C (or possibly) - There is a Makefile and it contains the	GiT repository. n ASM). usual rules.		
Sources Preliminaries Preliminaries First, check that the following is true: - Something has been uploaded to the orange of the project is written in C (or possibly) - There is a Makefile and it contains the	GiT repository. n ASM). usual rules. ce ends. You are still, however,		
Sources Preliminaries Preliminaries First, check that the following is true: - Something has been uploaded to the orange of the project is written in C (or possibly) - There is a Makefile and it contains the	GiT repository. n ASM). usual rules. ce ends. You are still, however,		
Sources Preliminaries Preliminaries First, check that the following is true: - Something has been uploaded to the orange of the project is written in C (or possibly) - There is a Makefile and it contains the	GiT repository. n ASM). usual rules. ce ends. You are still, however,	× No	

Utilization		
Running with arguments		
Before running the program, run the following command: $\ensuremath{`echo"int\ main(void) \{ printf(\"Hello\n\"); return 0; \}" > hello.c; gcc -m64 -o hello hello.c`}$ Then run the program with the 'hello' binary passed as the first parameter. There should now be a new file named 'woody'.		
	imesNo	
Usage		
Basic Usage		
Run 'woody'. The program should first output "WOODY", followed by a newline and "Hello".		
If that is not the case, the defence ends.		
	imesNo	
Advanced Usage		
Run the following command: `objdump -S woody > woody.obj ; objdump -S hello > hello.obj`		
You may now diff (hello.obg and woody.obj) with your favorite software and see that there are many differences.		
If that is not the case, the defence ends.		
⊗ Yes	imesNo	
Algorithm		
Explanation		

Note by a scale of how clear the algorithm is to the corrector after an explanation from the student. Understand how the algorithm can be improved. What pros and cons come with use of the algorithm. A maximum score should only be given if the corrector completely understands the algorithm after the explanation (or at the very least if the corrector is 100% certain that the student completely understands what they are talking about). Rate it from 0 (failed) through 5 (excellent) Complexity Note by a scale of difficulty/effectiveness of the chosen algorithm. The student must explain and justify their choice of alorithm. A simple ROT is not considered a complex algorithm! A maximum score should only be given in the case of a complex algorithm that creates a unique random key generated at runtime of the main executable. Rate it from 0 (failed) through 5 (excellent) **Error Management Error Management**

Before running the program, run the following command:

`echo "int main(void) { printf(\"Hello\n\"); return 0;}" > hello.c; gcc -m32 -o hello hello.c`

Then run the program with the 'hello' binary passed as the first parameter.

The program should either display an explicit error message, or in the case of a bonus, work as intended without any issues.

The corrector may then run the program with as many error cases as they can think of (invalid file, invalid path, not enough permissions, etc...).





woody_woodpacker bonus

Only grade the bonus section if ALL previous cases are PERFECT.

Bonus

All bonuses must be 100% functional and useful/convenient to be counted.

Examples of valid bonuses:

- 32bit support.
- Custom keys.
- Optimization of the algorithm used by the assembler.
- Support of multiple binary formats (PE, macho, ...).
- Compression of the binary.

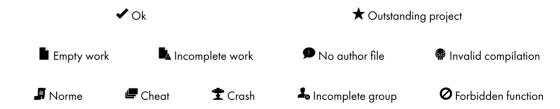
- ..



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

Ratings

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



Conclusion	
Leave a comment on this evaluation	
	/2
Preview!!!	

General term of use of the site (https://signin.intra.42.fr/legal/terms/6)

Privacy policy
(https://signin.intra.42.fr/legal/terms/5)

Legal notices
(https://signin.intra.42.fr/legal/terms/3)

Declaration on the use of cookies (https://signin.intra.42.fr/legal/terms/2)

Terms of use for video surveillance
(https://signin.intra.42.fr/legal/terms/1) (ht

(https://sigr