Assessment for vdepeshko's group

https://:@gitlab.ucode.world:8443/unit-factory-kyiv/uls/vdepeshko-5.git

Media materials

General

1. The goal is to share the experience with `assessor` and `defenders` about this challenge.
2. You have to check really carefully defending team's code in the specified repository. Team's knowledge depends on your evaluation.
3. Clone the repository.
4. You must verify the correctness of the challenge according to the `Auditor` rules. If at least one rule is violated, you must indicate this.
5. Files must be in the corresponding directory with names as specified in the story, if it is not true you must indicate it.
6. Be rigorous and honest, use the power of p2p and your brain.
7. Read the docs about p2p and defenses, if you have any disputes.
8. You must use the following flags to compile: `clang -std=c11 -Wall -Wextra -Werror -Wpedantic`. If the challenge does not compile you must indicate it.
9. You must check available work on the repository. Only files from SUBMIT part of the story must be assessed. You must indicate it if there are files that wasn't mentioned in the SUBMIT part.
10. There are 4 different types of questions in the protocol. You must check them according to the rules below:
    - `Binary a.k.a. true-false` - you must mark as `True`, only if everything works OK according to the questions, but if something fails at least one case, you must mark as `False`;
    - `Range 0-10` - you must add points strictly according to the instructions in the question;
    - `Label` - you must select a specific label according to the case detected during the whole assessment;
    - `Comment` - leave a descriptive and understandable comment according to the question.
11. Evaluation is carried out only in the presence of all members of the defense. Postpone assessment until all defense participants can come together.
12. Gain the knowledge and share your knowledge during the assessment.

Assign label below

Choose the first option that came up.

Help defending team to understand mistakes, discuss challenge in details, exchange of your knowledge.

Repository	Auditor	Libmx	Compiling	Memory leaks	Crashing	Extra files	Cheat	OK

If none of the items above are suitable.

Main

This is always what is clearly stated in the story. The assessment should be set honestly in accordance with the challenge.

Everything should work correctly, without runtime, compilation or logical errors.

The real assessment is much more valuable than overestimated or underestimated marks.

Error management

A very important part of software development is error management.

You always should consider during development that program/function can receive invalid input data.

Check for invalid command-line arguments.

The program must show error handling according to original `ls`.

Also do not forget to check the return value of the program.

You can do it by using `echo $?`, which shows the exit status of the previously executed command.

```
>ls foo
...
>echo $?
1
>./uls foo
...
>echo $?
1
```

Without flags

Compare `uls` work with the standard command `ls`.

Check without parameters, then with file and then with the directory.

Flag -l

Check whether `./uls -l` works correctly with files, directories, sockets etc.

Compare output with the standard command `ls -l`.

```
>./uls -l /
...
>./uls -l /dev/null
...
>./uls -l /usr/bin
...
```

What about links?

Links to the directory.

Try two simple commands with link to the directory:

command #1: `ls {link}`

output: The command has to show all files which exist in the directory.

command #2: `ls -l {link}`

output: The command has to show the information about link.

How does it work?

Extended file attributes

Check whether extended file attributes are implemented?

Is there `@` character next to the file permissions?

```
>./uls -l /
...
```

Access-control lists

Check whether access-control lists (ACL) are implemented?

Is there `+` character next to the file permissions?

```
>./uls -l /
...
```

Multi-column output

Does the solution work correctly with a multi-column output format `./uls` without flags.

Test with the directory containing enough files/dirs for multi-column output.

Try with different terminal window width.

Compare output with the standard command `ls`.

Flag -R

Check whether `./uls -R`.

Compare output with the standard command `ls -R`.

```
>./uls -R
...
>./uls -R /dev/null
...
>./uls -R /usr/bin
...
```

Flags -a, -A, -f

Check whether at least one of the `-a, -A, -f` flags has been implemented.

Test with different files/directories and compare output with the standard command `ls`.

Flags -Ra, -RA

Now check if the program works correctly with these combination of flags `./uls -Ra` or `./uls -RA`.

Compare output with the standard command `ls`.

Flag -G

Check whether `./uls -G` works correctly with files, directories, sockets etc.

Compare output with the standard command `ls -G`.

Does it look cute?

Flag combinations

Check whether flags that only make sense in combination with other flags are implemented. Add 2 points for each flag that works correctly.

For example try:

`ls -lh`, `ls -le`, `ls -lT`, `ls -l@`, etc.

0

Sorting flags

Check whether these `-r`, `-t`, `-c`, `-S`, `-u` etc. flags are implemented.

Add 2 points for each flag that works correctly.

0

Easy flags

Check whether these `-1`, `-C`, `-F`, `-m`, `-p` flags are implemented.

Add 2 points for each flag that works correctly.

0

Other flags

Were any additional flags implemented that were not marked above?

Add 2 points for each flag that works correctly.

0

Speed

Compare the speed of execution of your program with the original one. Use command `time`.

Test with big enough directory tree. Use flag `-R`.

```
>time ./uls -lR /Users/{xlogin}
...
>time ls -lR /Users/{xlogin}
...
```

Calculate `100 * ls_total_time / uls_total_time = speed_ratio`.

`Max` rate — `speed_ratio = 100`, `Min` rate — `speed_ratio = 0`.

0

Share

Have the team shared the solution with the world?

Check if they posted solution on GitHub/GitLab/BitBucket etc. Perhaps they wrote a post in any of the social networks where they talked about the challenge.

Have they written an article? If at least one of these statements is true, then rate it.

Reflection

This part is very important! You must evaluate how truly team that you assess understand the process of learning.

Does the team rightly evaluate its progress? Do students understand what they have learned during the challenge?

Talk with the defending team, discuss answers in the reflection protocol.

Evaluation

Use the slider to rate the responses.

Max rate if answers are detailed and they clearly reveal the correct essence, which corresponds to the theme of the task, Min rate - otherwise.

Be careful, low marks it's ok, study isn't always easy!

0

Open student's Reflection

Feedback

Your feedback about the evaluation.

Comment

Leave a comment on this evaluation.

Comment

Finish Assessment