Version: 1.0

Selection

Shell-Searching-And-Finding

Summary

Learn how to search, filter, and manipulate files using shell scripting and Unix command-line utilities.

#Shell #Unix #Find

Intellectual Property Disclaimer

All content presented in this training module, including but not limited to texts, images, graphics, and other materials, is protected by intellectual property rights held by Association 42.

Terms of Use:

- **Personal use:** You are permitted to use the contents of this module solely for personal purpose. Any commercial use, reproduction, distribution, modification, or public display is strictly prohibited without prior written permission from Association 42.
- **Respect for Integrity:** You must not alter, transform, or adapt the content in any way that could harm its integrity.

Protection of Rights:

Any violation of these terms constitutes an infringement of intellectual property rights and may result in legal action. We reserve the right to take all necessary measures to protect our rights, including but not limited to claims for damages.

For any questions regarding the use of the content or to obtain authorization, please contact: `legal@42.fr`

Contents

Chapter 1

Instructions

- Only this page will serve as a reference: do not trust rumors.
- Watch out! This document could potentially change up until submission.
- Make sure you have the appropriate permissions on your files and directories.
- You have to follow the submission procedures for all your exercises.
- Your exercises will be checked and graded by your fellow classmates.
- Additionally, your exercises will be checked and graded by a program called Moulinette.
- Moulinette is very meticulous and strict in its evaluation of your work. It is entirely automated, and there is no way to negotiate with it. So, to avoid bad surprises, be as thorough as possible.
- Moulinette is not very open-minded. It won't try to understand your code if it doesn't adhere to the Norm. Moulinette relies on a program called `norminette` to check if your files respect the norm. TL;DR: it would be foolish to submit work that doesn't pass `norminette`'s check.
- These exercises are carefully laid out by order of difficulty - from easiest to hardest. We `will not` consider a successfully completed harder exercise if an easier one is not perfectly functional.
- Using a forbidden function is considered cheating. Cheaters get -42, and this grade is non-negotiable.
- You'll only have to submit a main() function if we ask for a program.
- Moulinette compiles with these flags: -Wall -Wextra -Werror, and uses `cc`.
- If your program doesn't compile, you'll get `0`.
- You cannot leave any additional files in your directory other than those specified in the subject.
- Got a question? Ask your peer on the right. Otherwise, try your peer on the left.
- Your reference guide is called `Google / man / the Internet / ...`.
- Check out the Slack Piscine.

- Examine the examples thoroughly. They could very well call for details that are not explicitly mentioned in the subject...
- By Odin, by Thor! Use your brain!!!

Do not forget to add the *standard 42 header* in each of your .c/.h files. Norminette checks its existence anyway!

Norminette must be launched with the *-R CheckForbiddenSourceHeader* flag. Moulinette will use it too.

Context

The C Piscine is intense. It's your first big challenge at 42 — a deep dive into problem-solving, autonomy, and community.

During this phase, your main objective is to build your foundation — through struggle, repetition, and especially **peer-learning** exchange.

In the AI era, shortcuts are easy to find. However, it's important to consider whether your AI usage is truly helping you grow — or simply getting in the way of developing real skills.

The Piscine is also a human experience — and for now, nothing can replace that. Not even AI.

For a more complete overview of our stance on AI — as a learning tool, as part of the ICT curriculum, and as a growing expectation in the job market — please refer to the dedicated FAQ available on the intranet.

Main message

- 👉 Build strong foundations without shortcuts.
- 👉 Really develop tech & power skills.
- 👉 Experience real peer-learning, start learning how to learn and solve new problems.
- 👉 The learning journey is more important than the result.
- 👉 Learn about the risks associated with AI, and develop effective control practices and countermeasures to avoid common pitfalls.

Learner rules:

- You should apply reasoning to your assigned tasks, especially before turning to AI.
- You should not ask for direct answers to the AI.
- You should learn about 42 global approach on AI.

Phase outcomes:

Within this foundational phase, you will get the following outcomes:

- Get proper tech and coding foundations.
- Know why and how AI can be dangerous during this phase.

Comments and example:

- Yes, we know AI exists — and yes, it can solve your activities. But you're here to learn, not to prove that AI has learned. Don't waste your time (or ours) just to demonstrate that AI can solve the given problem.
- Learning at 42 isn't about knowing the answer — it's about developing the ability to find one. AI gives you the answer directly, but that prevents you from building your own reasoning. And reasoning takes time, effort, and involves failure. The path to success is not supposed to be easy.
- Keep in mind that during exams, AI is not available — no internet, no smartphones, etc. You'll quickly realise if you've relied too heavily on AI in your learning process.
- Peer learning exposes you to different ideas and approaches, improving your interpersonal skills and your ability to think divergently. That's far more valuable than just chatting with a bot. So don't be shy — talk, ask questions, and learn together!
- Yes, AI will be part of the curriculum — both as a learning tool and as a topic in itself. You'll even have the chance to build your own AI software. In order to learn more about our crescendo approach you'll go through in the documentation available on the intranet.

✓ Good practice:

> I'm stuck on a new concept. I ask someone nearby how they approached it. We talk for 10 minutes — and suddenly it clicks. I get it.

✗ Bad practice:

> I secretly use AI, copy some code that looks right. During peer evaluation, I can't explain anything. I fail. During the exam — no AI — I'm stuck again. I fail.

Chapter 2

Foreword

Simple Bread Recipe

- **Ingredients:**
  - 500g strong white flour
  - 1 tsp salt
  - 7g sachet fast-action dried yeast
  - 300ml warm water
  - 1 tbsp olive oil
- **Instructions:**

1. Mix the flour, salt, and yeast in a large bowl.
2. Make a well in the center, pour in the warm water and olive oil, and mix until a soft dough forms.
3. Turn out the dough onto a lightly floured surface and knead for 10 minutes.
4. Place the dough in a clean bowl, cover with a damp tea towel, and let it rise in a warm place for 1 hour.
5. Shape the dough and place it on a baking sheet. Let it rise for another 30 minutes.
6. Preheat the oven to 220°C (200°C fan) and bake for 25-30 minutes until golden.

Chapter 3

Exercise 0: find_sh

	Exercise: 0	
find_sh.sh		
Directory: *ex0/*		
Files to Submit: `find_sh.sh`		
Authorized: `None`		

- Write a command line that searches for all filenames ending with '.sh' (without quotation marks) in the current directory and all its sub-directories. It should display only the filenames without the '.sh' extension.
- Example Output :

Terminal Output

```
?>./find_sh.sh | cat -e
find_sh$
file1$
file2$
file3$
?>
```

Do not trust any source of information; always conduct your own tests, checks, and verifications.

Chapter 4

Exercise 1: clean

	Exercise: 1	
Directory: *ex1/*		
Files to Submit: `clean`		
Authorized: `None`		

- In a file called `clean`, place the command line that will search for all files in the current directory and its sub-directories with a name ending in ~, or a name that starts and ends by #.
- The command line will display and delete all found files.
- Only one command is allowed: no use of `';'` or `'&&'` or other tricks.

Refer to the 'man find' documentation.

Collaboration is key to success.

Chapter 5

Exercise 2: count_files

	Exercise: 2	
	count_files.sh	
Directory: *ex2/*		
Files to Submit: `count_files.sh`		
Authorized: `None`		

- Write a command line that counts and displays the number of regular files and directories in the current directory and all its sub-directories, including the starting directory '.'.
- Example Output :

Terminal Output

```
?>./count_files.sh | cat -e
42$
?>
```

Failure is a part of your learning journey.

Chapter 6

Submission and Peer Evaluation

Submit your assignment in your `Git` repository as usual. Only the work in your repository will be evaluated during the defense. Make sure to double-check the names of your files to ensure they are correct.

You need to submit only the files requested by this project's subject.