C Piscine

Shell 01

Summary: *This document contains the instructions for Shell module 01 of the C Piscine @ 42*

Version: 8.1

Contents

Chapter I

Instructions

- These exercises are carefully arranged in order of difficulty, from easiest to hardest. We <u>will not</u> consider a successfully completed harder exercise if an easier one is not perfectly functional.

- Ensure that you have the appropriate permissions on your files and directories.

- You must follow the **submission procedures** for every exercise.

- 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 extremely meticulous and strict in its evaluation. It is entirely automated, and there is no way to negotiate with it. To avoid unpleasant surprises, be as thorough as possible.

- Shell exercises must be executable with `/bin/sh`.

- You <u>must not</u> leave any additional files in your directory other than those specified in the assignment.

- Have a question? Ask the peer on your right. If not, try the peer on your left.

- Your reference guide is called **Google / man / the Internet / ...**

- Examine the examples carefully. They may contain details that are not explicitly mentioned in the assignment.

Chapter II

AI Instructions

● 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 projects. 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 III

Foreword

Here's what *Wikipedia* says about otters:

The European otter (Lutra lutra), also known as the Eurasian otter, Eurasian
river otter, common otter and Old World otter, is a European and Asian member
of the Lutrinae or otter subfamily, and is typical of freshwater otters.

The European otter is a typical species of the otter subfamily. Brown above
and cream below, these long, slender creatures are well-equipped for their
aquatic habits. Its bones show osteosclerosis, increasing their density to
reduce buoyancy.
This otter differs from the North American river otter by its shorter neck,
broader visage, the greater space between the ears and its longer tail.

However, the European otter is the only otter in its range, so it cannot be
confused for any other animal. Normally, this species is 57 to 95 cm (23-37 in)
long, not counting a tail of 35-45 cm (14-18 in).
The female is shorter than the male.

The otter's average body weight is 7 to 12 kg (15.4-26.4 lbs), although
occasionally a large old male may reach up to 17 kg (37 lbs).
The record-sized specimen, reported by a reliable source but not verified,
weighed over 24 kg (53 lbs).

The European otter is the most widely distributed otter species, its range
including parts of Asia and Africa, as well as being spread across Europe.
Though currently believed to be extinct in Liechtenstein and Switzerland,
they are now very common in Latvia, along the coast of Norway, and across
Great Britain, especially Shetland, which holds 12% of the UK's breeding population.
Ireland has the highest density of Eurasian otters in Europe.
In Italy, they can be found in southern parts of the peninsula.
The South Korean population is endangered.

Otters are cute.

Chapter IV

Exercise 00: Exam

	Exercise 00
	Exam

- During the week, you will be able to sign up for Friday's exam in the agenda, don't forget!

- You must also register for the **Exam00** project.

- Double-check that you are registered for both the exam event and the project!

- Triple-check that you are definitely registered for both the exam event and the project, yes, both!

Chapter V

Exercise 01: print_groups

	Exercise 01
	print_groups.sh
Turn-in directory: *ex01/*	
Files to turn in: `print_groups.sh`	
Allowed functions: `None`	

- Write a command line that displays the list of groups the user (defined in the environment variable `FT_USER`) belongs to.

- The output should be comma-separated, without spaces.

- Examples:

    ○ for `FT_USER=bocal`:

    ```
    $>./print_groups.sh
    bocal,adm,cdrom,sudo,dip,plugdev,lxd,lpadmin,libvirt$>
    ```

    ○ for `FT_USER=daemon`:

    ```
    $>./print_groups.sh
    daemon,bin$>
    ```

> - `man id`
>
> - Get inspired by others, but do the work yourself!

Chapter VI

Exercise 02: find__sh

	Exercise 02
	find__sh.sh
Turn-in directory: *ex02/*	
Files to turn in: `find_sh.sh`	
Allowed functions: `None`	

- Write a command line that searches for all files ending with `.sh` in the current directory and all subdirectories.

- The output should display only the file names without the `.sh` extension.

- Example output:

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

Don't blindly trust sources-always test, verify, and validate your results yourself!

Chapter VII

Exercise 03: count_files

	Exercise 03

count_files.sh
Turn-in directory: *ex03/*
Files to turn in: `count_files.sh`
Allowed functions: `None`

- Write a command line that counts and displays the total number of regular files and directories in the current directory and all its subdirectories.

- The count should include "." (the starting directory).

- Example output:

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

Failure is part of your learning journey-keep testing and improving!

Chapter VIII

Exercise 04: MAC

	Exercise 04
	MAC.sh
Turn-in directory: *ex04/*	
Files to turn in: `MAC.sh`	
Allowed functions: `None`	

- Write a command line that displays your machine's MAC addresses, with each address followed by a line break.

- `man ifconfig`

- `Collaboration is key to success!`

Chapter IX

Exercise 05: Can you create it ?

	Exercise 05
	Can you create it ?
Turn-in directory: *ex05/*	
Files to turn in: `"\?$*'MaRViN'*$?\"`	
Allowed functions: `None`	

- Create a file containing **only** "42", and **nothing else**.

- The file name must be:

```
"\?$*'MaRViN'*$?\"
```

- Example output:

```
$>ls -lRa *MaRV* | cat -e
-rw---xr-- 1 75355 32015 2 Oct 2 12:21 "\?$*'MaRViN'*$?\"$
$>
```

Milestone Achieved, Keep Going!

You've completed the mandatory exercises for this project. Now, you have a choice:

- Continue with the **optional exercises** to explore more.

- Move on to your **next project**.

Both paths will introduce you to useful concepts. Consider the following before making your decision:

- Your first exam, as well as the end-of-week rush, will focus on C programming. It might therefore be useful to gain experience in this field beforehand. (You'll learn more about the rush soon).

- Your performance in this Piscine is evaluated on multiple factors:

  - Project completion is one aspect.

  - Overall progress through the full list of Piscine projects is another. Choose wisely to maximize your results.

- You can retry the same project in a few days or weeks until the end of the Piscine.

- Staying in sync with your peers promotes better collaboration.

Chapter X

Exercise 06: Skip

	Exercise 06
	skip.sh
Turn-in directory: *ex06/*	
Files to turn in: `skip.sh`	
Allowed functions: `None`	

- Write a command line that executes `ls -l` but displays only every second line, starting from the first line.

- Example output:

```
$>ls -l | cat -e
total 4$
-rw-rw-r-- 1 eagle eagle ** ** 15 15:46 skip.sh$
-rw-rw-r-- 1 eagle eagle ** ** 15 15:41 tata$
-rw-rw-r-- 1 eagle eagle ** ** 15 15:41 titi$
-rw-rw-r-- 1 eagle eagle ** ** 15 15:41 toto$
-rw-rw-r-- 1 eagle eagle ** ** 15 15:41 tutu$
$>
```

```
$>./skip.sh | cat -e
total 4$
-rw-rw-r-- 1 eagle eagle ** ** 15 15:41 tata$
-rw-rw-r-- 1 eagle eagle ** ** 15 15:41 toto$
$>
```

Git push regularly!

Chapter XI

Exercise 07: r__dwssap

	Exercise 07
	r__dwssap.sh
Turn-in directory: *ex07/*	
Files to turn in: `r_dwssap.sh`	
Allowed functions: `None`	

- Write a command line that processes the output of `cat /etc/passwd` with the following modifications:

    ○ Remove comments.

    ○ Keep every other line, starting from the second line.

    ○ Reverse each login name.

    ○ Sort the results in reverse alphabetical order.

    ○ Keep only logins between the environment variables `FT_LINE1` and `FT_LINE2` (inclusive).

    ○ Join them in a single line, separated by ", ".

    ○ End the output with a "."

- Example Output (for lines 7 to 15):

```
$> ./r_dwssap.sh
sstq_, sorebrek_brk_, soibten_, sergtsop_, scodved_, rlaxcm_, rgmecived_, revreswodniw_,
    revressta_.$>
```

> - Follow the steps in the exact order given!
>
> - Did you check with your left-side neighbor?

Chapter XII

Exercise 08: add_chelou

	Exercise 08
	add_chelou.sh

Turn-in directory: *ex08/*
Files to turn in: `add_chelou.sh`
Allowed functions: `None`

- Write a command line that:

  - takes numbers from variables `FT_NBR1`, in `'\"?!` base and `FT_NBR2`, in `mrdoc` base.

  - Displays the sum of both in the `gtaio luSnemf` base.

- Example 1:
  ```
  FT_NBR1=\'?"\"'\
  FT_NBR2=rcrdmddd
  ```

- The result is:
  ```
  Salut
  ```

- Example 2 :
  ```
  FT_NBR1=\"\"!\"\"!\"\"!\"\"!\"\"!\"\"
  FT_NBR2=dcrcmcmooododmrrrmorcmcrmomo
  ```

- The result is :
  ```
  Segmentation fault
  ```

If you're stuck, randomly ask someone in your cluster for help!

Chapter XIII

Submission and peer-evaluation

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

> ⚠ You must submit only the files explicitly required by the project instructions.