C Bootcamp

Day 01

Staff WeThinkCode_ pedago@wethinkcode.co.za

Summary: This document is the subject for Day01 of the C Bootcamp @ WeThinkCode_.

Contents

Chapter I

Instructions

- Only this page will serve as reference; do not trust rumors.
- Watch out! This document could potentially change up to an hour before submission.
- These exercises are carefully laid out by order of difficulty - from easiest to hardest. We `will not` take into account a successfully completed harder exercise if an easier one is not perfectly functional.
- Make sure you have the appropriate permissions on your files and directories.
- You have to follow the submission procedures for every exercise.
- Your exercises will be checked and graded by your fellow classmates.
- On top of that, 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 if you want to avoid bad surprises, be as thorough as possible.
- Exercises in Shell must be executable with /bin/sh.
- You cannot have any additional file in your directory than those specified in the subject.
- Got a question? Ask your peer on the right. Otherwise, try your peer on the left.
- Your reference guides are called `Google / man / the Internet / ....`
- 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!!!

Chapter II

Foreword

Here's what `Wikipedia` has to say 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, where 12% of the UK breeding population
exist. 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 III

Introduction

For this day01 you will have to create shell scripts. Yesterday we were asking you to put commands in a file. If this file is executable it becomes a program.

Take a look at the `chmod` command and check out how to make a file be executable.

May the code always be in your favour.

Chapter IV

Exercise 00 : Exam

	Exercise 00
Exam	
Turn-in directory : *ex*00/	
Files to turn in : `Exam file`	
Allowed functions : `None`	
Notes : `n/a`	

- On the Intranet's dashboard, in the Agenda section there is a red event for Friday's exam, click on it and `Subscribe` for it.

- You also have to register for the Exam00 project. Like yesterday you need to go in the list of projects and register for it.

- Make sure you've registered for the exam correctly (the event AND the project!).

- Make sure you've made sure you've registered for the exam correctly (the event AND the project! Yep, both!).

Commit and push a file with the name of the project and the name of the event that you just registered to separated by a comma in a file called Exam.

Chapter V

Exercise 01 : Push !

	Exercise : 01
Push !	
Notes : `n/a`	

```
Ensure that you did commit and push the file of the previous exercise
before moving to the next. This will be checked during peer-2-peer.
```

Chapter VI

Exercise 02 : Who am I ?

	Exercise 02
Who am I ?	
Turn-in directory : *ex*02/	
Files to turn in : `who_am_i.sh`	
Allowed functions : `None`	
Notes : `n/a`	

Yesterday you learned about Kerkebos ticket, it's time to find out who you are!

As you might have guessed, every student at school is on LDAP: some sort of rudimentary digital phonebook - for those who remember what a phonebook is...

- Create a script called `who_am_i.sh`, that will return only the value of the distinguished name.
- For example, with a `test` Kerberos ticket:

```
%>sh who_am_i.sh
uid=test,ou=2013,ou=people,dc=42,dc=fr
%>
```

All commands to communicate with LDAP start with ... ldap.

The first four lines (starting with SASL) will never be taken into account for mysterious reasons you'll understand later on.

Chapter VII

Exercise 03 : Push !

	Exercise : 03
Push !	
Notes : `n/a`	

Ensure that you did commit and push the file of the previous exercise before moving to the next. This will be checked during peer-2-peer.

Chapter VIII

Exercise 04 : print_groups

	Exercise 04
print_groups.sh	
Turn-in directory : *ex*04/	
Files to turn in : `print_groups.sh`	
Allowed functions : `None`	
Notes : `n/a`	

For this exercise you will have to look for something called `environment variables` how to create them and how to use them.

- Create a script called `print_groups.sh` that will display the list of groups for which the login, contained in the environment variable FT_USER, is a member. The groups will be separated by a commas without spaces.

- Examples :

  - for FT_USER=nours, the result is "god,root,admin,master,nours,bocal" (without quotation marks)

```
$>./print_groups.sh
god,root,admin,master,nours,bocal$>
```

  - for FT_USER=daemon, the result is "daemon,bin" (without quotation marks)

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

`man groups`

Chapter IX

Exercise 05 : Push !

	Exercise : 05
Push !	
Notes : `n/a`	

`Ensure that you did commit and push the file of the previous exercise before moving to the next. This will be checked during peer-2-peer.`

Chapter X

Exercise 06 : find_sh

	Exercise 06
find_sh.sh	
Turn-in directory : *ex*06/	
Files to turn in : `find_sh.sh`	
Allowed functions : `None`	
Notes : `n/a`	

- Create a script called `find_sh.sh` that searches for all file names that end with `.sh` in the current directory and all its sub-directories. It should display only the file names without the `.sh`.

- Example of output :

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

Chapter XI

Exercise 07 : Push !

	Exercise : 07
Push !	
Notes : n/a	

Ensure that you did commit and push the file of the previous exercise before moving to the next. This will be checked during peer-2-peer.

Chapter XII

Exercise 08 : count_files

	Exercise 08
count_files.sh	
Turn-in directory : *ex*08/	
Files to turn in : `count_files.sh`	
Allowed functions : `None`	
Notes : `n/a`	

- Create a script called `count_files.sh` that counts and displays the number of regular files and directories in the current directory and all its sub-directories. It should include ".", the starting directory.
- Example of output :

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

Chapter XIII

Exercise 09 : Push !

	Exercise : 09
Push !	
Notes : `n/a`	

`Ensure that you did commit and push the file of the previous exercise before moving to the next. This will be checked during peer-2-peer.`

Chapter XIV

Exercise 10 : MAC

	Exercise 10
MAC.sh	
Turn-in directory : *ex*10/	
Files to turn in : `MAC.sh`	
Allowed functions : `None`	
Notes : `n/a`	

- Create a script called `MAC.sh` that displays your machine's MAC addresses. Each address must be followed by a line break.

`man ifconfig`

Chapter XV

Exercise 11 : Push !

	Exercise : 11
Push !	
Notes : `n/a`	

`Ensure that you did commit and push the file of the previous exercise before moving to the next. This will be checked during peer-2-peer.`

Chapter XVI

Exercise 12 : Can you create it ?

	Exercise 12
Can you create it ?	
Turn-in directory : *ex*12/	
Files to turn in : `"\?$*'KwaMe'*$?\"`	
Allowed functions : `None`	
Notes : `n/a`	

- Create a file containing only "42", and NOTHING else.
- Its name will be :

```
"\?$*'KwaMe'*$?\"
```

- Example :

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

Chapter XVII

Exercise 13 : Push !

	Exercise : 13
Push !	
Notes : `n/a`	

`Ensure that you did commit and push the file of the previous exercise before moving to the next. This will be checked during peer-2-peer.`

Chapter XVIII

Exercise 14 : Skip

	Exercise 14
skip.sh	
Turn-in directory : *ex*14/	
Files to turn in : `skip.sh`	
Allowed functions : `None`	
Notes : `n/a`	

- Create a script called `skip.sh` that displays every other line for the command `ls -l`, starting from the first line.

Chapter XIX

Exercise 15 : Push !

	Exercise : 15
Push !	
Notes : `n/a`	

Ensure that you did commit and push the file of the previous exercise before moving to the next. This will be checked during peer-2-peer.

Chapter XX

Exercise 16 : r_dwssap

	Exercise 16
r_dwssap.sh	
Turn-in directory : *ex*16/	
Files to turn in : `r_dwssap.sh`	
Allowed functions : `None`	
Notes : `n/a`	

- Create a script called `r_dwssap.sh` that displays the output of the `cat /etc/passwd` command, removing comments, every other line starting from the second line, reversing each login, sorted in reverse alphabetical order, and keeping only logins between the environment variables FT_LINE1 and FT_LINE2 included, and they must separated by ", " (a comma and a spacewithout quotation marks), and the output must end with a ".".

- Example: Between lines 8 and 16, the result should be something like this:

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

`Rigorously follow the order indicated in the instructions.`

Chapter XXI

Exercise 17 : Push !

	Exercise : 17
Push !	
Notes : `n/a`	

`Ensure that you did commit and push the file of the previous exercise before moving to the next. This will be checked during peer-2-peer.`

Chapter XXII

Exercise 18 : strange_add

	Exercise 18
strange_add.sh	
Turn-in directory : *ex*18/	
Files to turn in : `strange_add.sh`	
Allowed functions : `None`	
Notes : `n/a`	

- Create a script called `strange_add.sh` that takes numbers from environment variables `FT_NBR1`, in `'\"?!` base, and `FT_NBR2`, in `mrdoc` base, and displays the sum of both in `gtaio luSnemf` base.
  - Example 1:

```
FT_NBR1=\'?"\"'\
FT_NBR2=rcrdmddd
```

  - The sum is:

```
Salut
```

  - Example 2:

```
FT_NBR1=\"\"!\"\"!\"\"!\"\"!\"\"!\"\"
FT_NBR2=dcrcmcmooodmrrrmorcmcrmomo
```

  - The sum is:

```
Segmentation fault
```

Obviously for this exercise you need to search for the different bases that are requested if you want to be able to succeed.

Chapter XXIII

Exercise 19 : Push !

	Exercise : 19
Push !	
Notes : `n/a`	

`Ensure that you did commit and push the file of the previous exercise before moving to the next. This will be checked during peer-2-peer.`