

CS 97 - Discussion 1F

Week 5

Midterm Review

Contents

1. **Logistic**
2. Strategy
3. Overview with Practice

1. Logistic

- CCLE: quiz
 - Try it: **Demo midterm (Test Logistics)**
- Timing:
 - **24-hr** window to take the exam: Feb 8 (Mon) 12 PM PST - Feb 9 (Tue) 12 PM PST
 - Once you start: **2 hrs** to complete the exam
 - You must start before Tuesday 10 AM for a full-time quiz
 - Professor will open a short Zoom session at Mon 12 PM to start the midterm instead of a normal lecture
- Resource
 - Test is open computer, open notes, open SEAS, open Google
 - **NO** collaboration / communication with others! Finish by yourself !!

1. Logistic - Questions?

- Piazza
 - Will be “locked down” to private posts to instructors
 - We will keep a pinned post on the top if there are common questions/answers that everyone should know
 - TA will be ensured to be online at certain hours:
 - Monday: 12 PM - 4 PM, 7 PM - 10 PM PST
 - Tuesday: 8 AM - 11 AM
 - We recommend taking the exam to overlap with these times in case you have questions
- Office hour:
 - I will still host the office hour from 9:30 PM to 10:30 PM
 - But I will only answer questions related to the logistic of the midterm
 - (Of course, I cannot directly tell you the answer 😊)

Contents

1. Logistic
- 2. Strategy**
3. Overview with Practice

2. Strategy

- Make use of the 2 hrs you have!
 - Check how many points (minutes) the exam says for each questions
 - Usually Qs with fewer points are okay to have shorter questions
 - Usually Qs with larger points will take longer time
 - Easy ones first:
 - First answer as many questions as you easily can
 - Then come back later to the hard ones
 - Don't worry about the perfect answers:
 - Try to move on and get as many points as you can
- For hard questions:
 - Try to focus more on the overall approach for more points, then come back for small details (which might be worth fewer points)
 - If Qs ask you to explain, try to focus more on concrete justification for your answer

2. Strategy

- Review your HWs/Labs
- Make use of the resources
 - Open reference pages for you to check out
 - Summarize some common linux shell commands
 - Doc / Syntax for shell scripting
 - Emac reference card
 - Doc / Syntax for Lisp
 - Doc / Syntax for python
 - Doc / Syntax for HTML
 - Doc / Syntax for Javascript
 - ...

Contents

1. Logistic
2. Strategy
- 3. Overview with Practice**

3. Overview

- Files, Shell and shell scripting
 - Process / Unix file system / POSIX
 - Basic shell commands: [man](#), [ls](#), [pwd](#), [cd](#), [mkdir](#), [rmdir](#), [echo](#), [cat](#), [cp](#), [mv](#), [ln](#), [rm](#), [chmod](#), [kill](#), [ps](#), [grep](#), [find](#)
 - Pattern matching, Regular Expression (Basic vs Extended)
 - Pipeline and in/out redirection
 - Scripting
- Files, editing
 - Emacs and basic commands
 - Emacs Lisp: Define and using functions
- Python
 - Variables, control flow, functions, classes, modules
- Client-server apps and user interfaces
 - HTML / JavaScript / React

3. Practice - Sample Midterm

1 (2 minutes). What is the likely typo in the following shell command, and why is it such a serious typo?

```
rm *.*[Oo][Uu][Tt] * .o *.a
```

- What does command `rm` do?
- What does `*` means here?
- What if I do `rm *`

3. Practice - Sample Midterm

1 (2 minutes). What is the likely typo in the following shell command, and why is it such a serious typo?

```
rm *.*[Oo][Uu][Tt] * .o *.a
```

- What does command `rm` do?
- What does `*` means here?
- What if I do `rm *`
- Extra space in between `*` and `.o`
- Will delete everything in the directory instead of deleting files `*.o`

3. Practice - Sample Midterm

2. A set of read/write/execute permissions on a file is called “sensible” if the owner has all the permissions of the group, and the group has all the permissions of others. For example, 551 (octal) is sensible, whereas 467 (octal) is not.

2a (2 minutes). Briefly explain why non-sensible permissions don't make much sense.

- Understand the definition of “sensible” permissions first!
- Permission: Owner \geq Groups \geq Others
- Who's the owner? Who's the groups? Who's the others?

3. Practice - Sample Midterm

2. A set of read/write/execute permissions on a file is called “sensible” if the owner has all the permissions of the group, and the group has all the permissions of others. For example, 551 (octal) is sensible, whereas 467 (octal) is not.

2a (2 minutes). Briefly explain why non-sensible permissions don't make much sense.

- Understand the definition of “sensible” permissions first!
- Permission: Owner \geq Groups \geq Others
- Who's the owner? Who's the groups? Who's the others?
- **Owner**: the person who is considered the primary owner of the file (the closest relation to the file)
- **Group**: a group of people who owns (connected to) the file in some way (but not as much as the owner) (the next closest relation to the file)
- **Other**: all users other than owner and group (the least relation to the file)
- People with less relation to the file should **NOT** have more permissions

3. Practice - Sample Midterm

2b (6 minutes). How many distinct sensible permissions are there? Explain.

- Understand the definition of “sensible” permissions first!
- Permission: Owner \geq Groups \geq Others

1. List all possibilities and count

If other's bits all 0: 000 (1 case)				
Possible Group bits	000	100, 010, 001	110, 011, 101	111
#Possible owner bits	All 8 cases	4 per case	2 per case	1
#sub-total	8	$3 \times 4 = 12$	$3 \times 2 = 6$	1
Total: $1 \times (8 + 12 + 6 + 1) = 27$				

1. If other's bits all 0: 000 (1 case)
2. If other has one bit set: 010, 100, 001 (3 cases)
3. If other has two bits set: 110, 011, 101 (3 cases)
4. If other has three bits set: 111 (1 cases)

1. 27
2. 27
3. 9
4. 1

Total: $27 + 27 + 9 + 1 = 64$

3. Practice - Sample Midterm

2b (6 minutes). How many distinct sensible permissions are there? Explain.

- Understand the definition of “sensible” permissions first!
- Permission: Owner \geq Groups \geq Others

2. Write a program, use your favorite programming language

3. Practice - Sample Midterm

4 (4 minutes). Give an example of how renaming a dangling symbolic link can transform it into a non-dangling symbolic link.

- What is symbolic link?
 - Symbolic link points to a file name
- What makes the symbolic dangling?
 - If the file name no longer exists
- How to turn it back to non-dangling?
 - Make the file exists!

```
❏ ~ /Desktop/cs97-test/midterm > echo "a" > badFile
❏ ~ /Desktop/cs97-test/midterm > ln -s badFile badSoftLink
❏ ~ /Desktop/cs97-test/midterm > ll
total 12K
-rw-rw-r-- 1 yuxing yuxing 78 Feb 4 18:04 6-dict.py
-rw-rw-r-- 1 yuxing yuxing 2 Feb 4 18:09 badFile
lrwxrwxrwx 1 yuxing yuxing 7 Feb 4 18:09 badSoftLink -> badFile
-rw-rw-r-- 1 yuxing yuxing 891 Feb 4 17:54 sensible_permission.py
❏ ~ /Desktop/cs97-test/midterm > rm badFile
❏ ~ /Desktop/cs97-test/midterm > ll
total 8.0K
-rw-rw-r-- 1 yuxing yuxing 78 Feb 4 18:04 6-dict.py
lrwxrwxrwx 1 yuxing yuxing 7 Feb 4 18:09 badSoftLink -> badFile
-rw-rw-r-- 1 yuxing yuxing 891 Feb 4 17:54 sensible_permission.py
❏ ~ /Desktop/cs97-test/midterm > echo "another a" > badFile
❏ ~ /Desktop/cs97-test/midterm > ll
total 12K
-rw-rw-r-- 1 yuxing yuxing 78 Feb 4 18:04 6-dict.py
-rw-rw-r-- 1 yuxing yuxing 10 Feb 4 18:09 badFile
lrwxrwxrwx 1 yuxing yuxing 7 Feb 4 18:09 badSoftLink -> badFile
-rw-rw-r-- 1 yuxing yuxing 891 Feb 4 17:54 sensible_permission.py
```

```
❏ ~ /Desktop/cs97-test/midterm > echo "another a" > badFile
❏ ~ /Desktop/cs97-test/midterm > ll
total 12K
-rw-rw-r-- 1 yuxing yuxing 78 Feb 4 18:04 6-dict.py
-rw-rw-r-- 1 yuxing yuxing 10 Feb 4 18:09 badFile
lrwxrwxrwx 1 yuxing yuxing 7 Feb 4 18:09 badSoftLink -> badFile
-rw-rw-r-- 1 yuxing yuxing 891 Feb 4 17:54 sensible_permission.py
❏ ~ /Desktop/cs97-test/midterm > rm badFile
❏ ~ /Desktop/cs97-test/midterm > ll
total 8.0K
-rw-rw-r-- 1 yuxing yuxing 78 Feb 4 18:04 6-dict.py
lrwxrwxrwx 1 yuxing yuxing 7 Feb 4 18:09 badSoftLink -> badFile
-rw-rw-r-- 1 yuxing yuxing 891 Feb 4 17:54 sensible_permission.py
❏ ~ /Desktop/cs97-test/midterm > echo "a" > anotherFile
❏ ~ /Desktop/cs97-test/midterm > ln -sf anotherFile badSoftLink
❏ ~ /Desktop/cs97-test/midterm > ll
total 12K
-rw-rw-r-- 1 yuxing yuxing 78 Feb 4 18:04 6-dict.py
-rw-rw-r-- 1 yuxing yuxing 2 Feb 4 18:10 anotherFile
lrwxrwxrwx 1 yuxing yuxing 11 Feb 4 18:10 badSoftLink -> anotherFile
-rw-rw-r-- 1 yuxing yuxing 891 Feb 4 17:54 sensible_permission.py
```

```
❏ ~ /Desktop/cs97-test/midterm > mkdir tmp
❏ ~ /Desktop/cs97-test/midterm > echo "test" > tmp/badFile
❏ ~ /Desktop/cs97-test/midterm > ln -s badFile badSoftLink
❏ ~ /Desktop/cs97-test/midterm > ll
total 24K
-rw-rw-r-- 1 yuxing yuxing 85 Feb 4 18:11 6-dict.py
-rw-rw-r-- 1 yuxing yuxing 30 Feb 4 18:13 7-sysmodules.py
-rwxrwxrwx 1 yuxing yuxing 134 Feb 4 18:21 8-shell.sh
-rw-rw-r-- 1 yuxing yuxing 2 Feb 4 18:10 anotherFile
lrwxrwxrwx 1 yuxing yuxing 7 Feb 4 22:46 badSoftLink -> badFile
-rw-rw-r-- 1 yuxing yuxing 891 Feb 4 17:54 sensible_permission.py
drwxrwxr-x 2 yuxing yuxing 4.0K Feb 4 22:45 tmp
❏ ~ /Desktop/cs97-test/midterm > mv badSoftLink tmp
❏ ~ /Desktop/cs97-test/midterm > ll tmp
total 4.0K
-rw-rw-r-- 1 yuxing yuxing 5 Feb 4 22:45 badFile
lrwxrwxrwx 1 yuxing yuxing 7 Feb 4 22:46 badSoftLink -> badFile
```


3. Practice - Sample Midterm

5 (8 minutes). Explain how to arrange for Emacs to treat C-t (i.e., control T) as a command that causes Emacs to issue a message like this:

It is now Tue Apr 27 10:30:34 2020.

in the echo area. The message should contain the current date and time.

- How did you add commands in Emacs in you HW?
 - Write a list function
 - Load that file to Emacs (`M-x load-file <RET> /path/to/your/file`)
- What we don't know? (Search it on Google, Emacs Reference Card,)
 - How to bind the function to a key?
 - `M-x global-set-key <RET> C-t print-time <RET>`
 - How to print the time?
 - `current-time-string`
 - `format-time-string`

```
(defun print-time ()  
  (interactive)  
  (message (concat "It is now " (current-time-string) ".")))  
  
(defun print-time ()  
  (interactive)  
  (message (format-time-string "It is now %a %b %e %T %Y.")))
```

3. Practice - Sample Midterm

6 (4 minutes). If a Python class C has a method M and a class variable V, what sort of thing does `C.__dict__.items()` return and why?

- `C.__dict__.items()` What's that?

- Write a program for this

```
1 class C:
2     ...V = 'someVar'
3
4     ...def M():
5         ...return 1
6
7 print(C.__dict__.items())
```

- Google it

- We see that `C.__dict__.items()` returns a `dict_items` object containing (among others) the entries
 - `'V'` → the value we set the class variable `v` to
 - `'M'` → <the function `C.M`>
- This is because the `__dict__` property on objects contains all of an object's writable attributes, which includes both its data members and methods.

3. Practice - Sample Midterm

7. By default, the Python expression `'sys.modules'` signals a `NameError`, but if you execute `'import sys'` first the expression does not signal that error.

7a (2 minutes). Briefly explain why not.

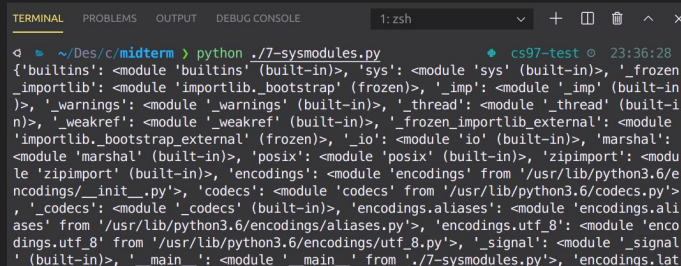
- Test on your computer.
- What does `import xxx` mean?
- There is no variable named `'sys'` that exists by default.
- Once the `import` statement is executed, however, the `sys` module is imported and bound to the identifier `'sys'`.

3. Practice - Sample Midterm

7b (4 minutes). Give an example of using the value of the expression 'sys.modules' to find out something about your Python process. Your example should explore at least one level past the value of sys.modules itself.

- Test on your computer or Google it.
- A dict of all imported modules in the current Python process
- Explore more!

```
1 import sys
2
3 print(sys.modules)
```



```
TERMINAL  PROBLEMS  OUTPUT  DEBUG CONSOLE  1: zsh  +  -  x
~/Des/c/midterm > python ./7-sysmodules.py  cs97-test 23:36:28
{'builtins': <module 'builtins' (built-in)>, 'sys': <module 'sys' (built-in)>, '_frozen_importlib': <module 'importlib._bootstrap' (frozen)>, '_imp': <module '_imp' (built-in)>, '_warnings': <module '_warnings' (built-in)>, '_thread': <module '_thread' (built-in)>, '_weakref': <module '_weakref' (built-in)>, '_frozen_importlib_external': <module 'importlib._bootstrap_external' (frozen)>, '_io': <module 'io' (built-in)>, '_marshal': <module 'marshal' (built-in)>, '_posix': <module 'posix' (built-in)>, '_zipimport': <module 'zipimport' (built-in)>, 'encodings': <module 'encodings' from '/usr/lib/python3.6/encodings/__init__.py'>, 'codecs': <module 'codecs' from '/usr/lib/python3.6/codecs.py'>, '_codecs': <module '_codecs' (built-in)>, 'encodings.aliases': <module 'encodings.aliases' from '/usr/lib/python3.6/encodings/aliases.py'>, 'encodings.utf_8': <module 'encodings.utf_8' from '/usr/lib/python3.6/encodings/utf_8.py'>, '_signal': <module '_signal' (built-in)>, '_main_': <module '_main_' from './7-sysmodules.py'>, 'encodings.la
```

3. Practice - Sample Midterm

8. Consider the following shell script.

```
#!/bin/sh
atom='[a-zA-Z0-9]+'
string='\"([^\"]|\\.)*\"'
word="($atom|$string)"
words="$word(\\.$word)*"
grep -E "$words" | grep ' '
```

8a (4 minutes). Explain briefly what this shell script does, from its user's viewpoint.

- Figure out what we know
 - atom = regex for alphanumeric character and 1 or more
 - string = ? looks so complicated
 - word = regex: atom or string
 - words = word + optional extra stuff
 - grep = works like a filter, ensure the output fits words with a space
- It prints any line that satisfy both of the following:
 - Contains a space
 - Contains at least one of:
 - An alphanumeric character (\$atom) or
 - A (possibly empty) sequence of (\$string)
- If you have more time, figure out what string is:
 - A (possibly empty) sequence of non-`\"` (literal double-quote) characters that is surrounded by `\"` (literal backslash followed by literal double-quote). If `\` (literal backslash) appears within the string, then it must start a well-formed escape sequence (it has another character that follows it) before the closing `\"` (\$string).

3. Practice - Sample Midterm

8b (3 minutes). How would the script's behavior change if you removed the '-E' from this script?

- Use `man grep` to see what does this option mean / Google it
- If we were to remove the '-E' flag, then POSIX Basic Regular Expression (BRE) is used instead of the Extended Regular Expression (ERE).
- This means that many meta-characters used, including the '(', '|', and ')' in \$word, are treated as normal characters to match rather than the start of group, disjunction, and end of a group, respectively.

3. Practice - Sample Midterm

8c (5 minutes). Modify the original script so that calls grep just once instead of twice, without changing the script's I/O behavior.

- Many ways to achieve that!

E.g.:

```
space=" "  
grep -E "$words$space|$space$words"
```