Gaulthier Gain & Kenichi Yasukata

INFO0940 Operating Systems

Project #1

Academic year 2019-2020

YOUR FIRST PROJECT

It is a Shell…

1. Not the same as previous years (so don't take old projects);
2. We provide a skeleton (we implemented the parsing and the interface logic);
3. You will understand process creation (a bit useful for the oral exam).

A classic Shell

It is a simple program that has a very basic logic:

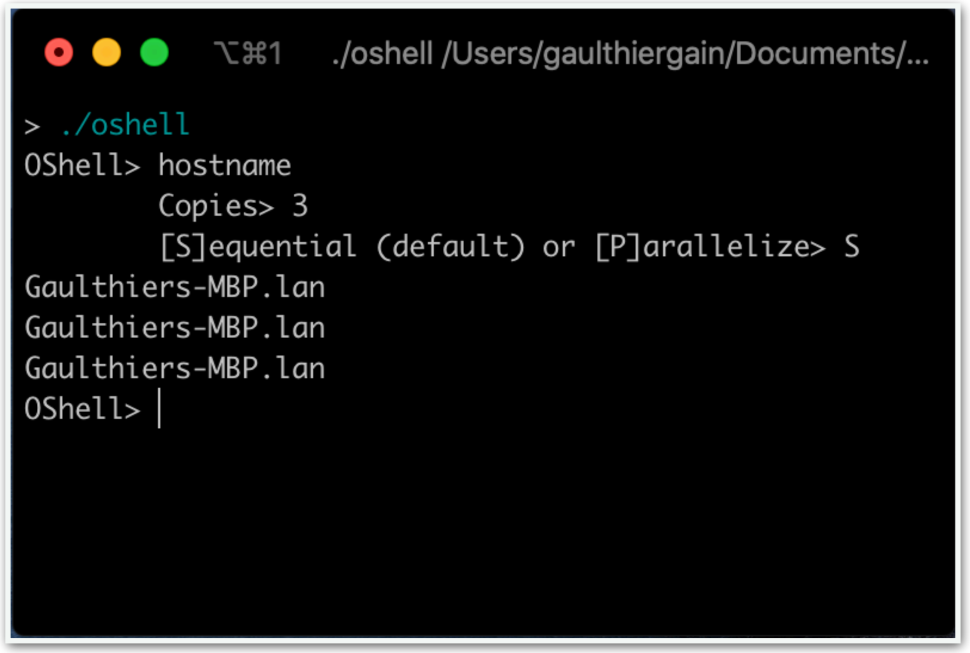
1. It loops;
2. It handles and parses commands by reading the standard input;
3. It then executes them.

THE OSHELL: MAIN FEATURES

Like a classical shell except that:

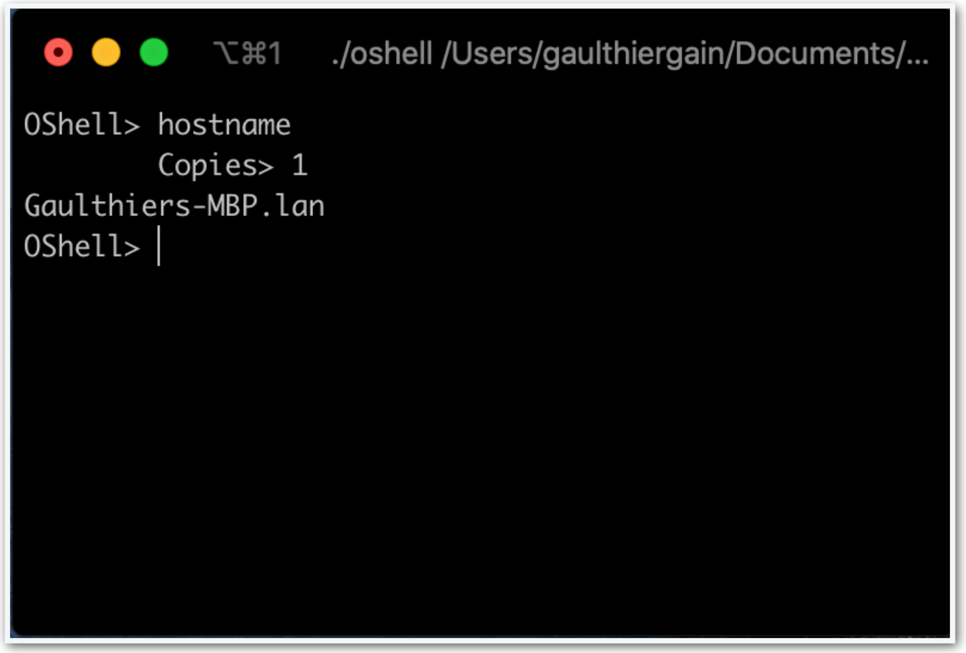
1. It can run several commands at once;
2. It handles parallel and/or sequential execution;
3. It maintains a list of process that were executed;
4. It has some built-in commands (`cd`, `memdump`, `loadmem`, ...);
5. It does **not** handle pipes (*e.g.*, `ls -al | wc`) and complex commands (*e.g.*, `sleep 3 && echo hello`).

The OShell:
1. Run several commands



```
> ./oshell
OShell> hostname
        Copies> 3
        [S]equential (default) or [P]arallelize> S
Gaulthiers-MBP.lan
Gaulthiers-MBP.lan
Gaulthiers-MBP.lan
OShell> 
```

3 executions



```
OShell> hostname
        Copies> 1
Gaulthiers-MBP.lan
OShell> 
```

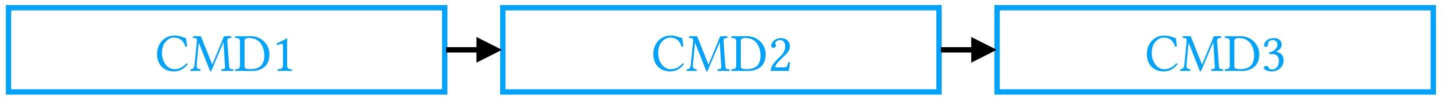
1 execution

`copies`: The number of copies of the program to execute. It must be an integer between **1** to **9**.

<u>**Note:**</u> If the number of copies equals one, the shell executes directly the command.

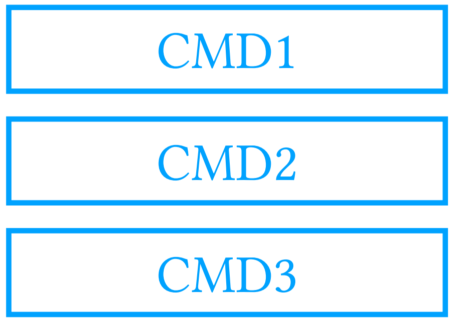
The OShell:
2. Parallel/Sequential Execution

Sequential Execution:



Hint: No need of IPC system V!

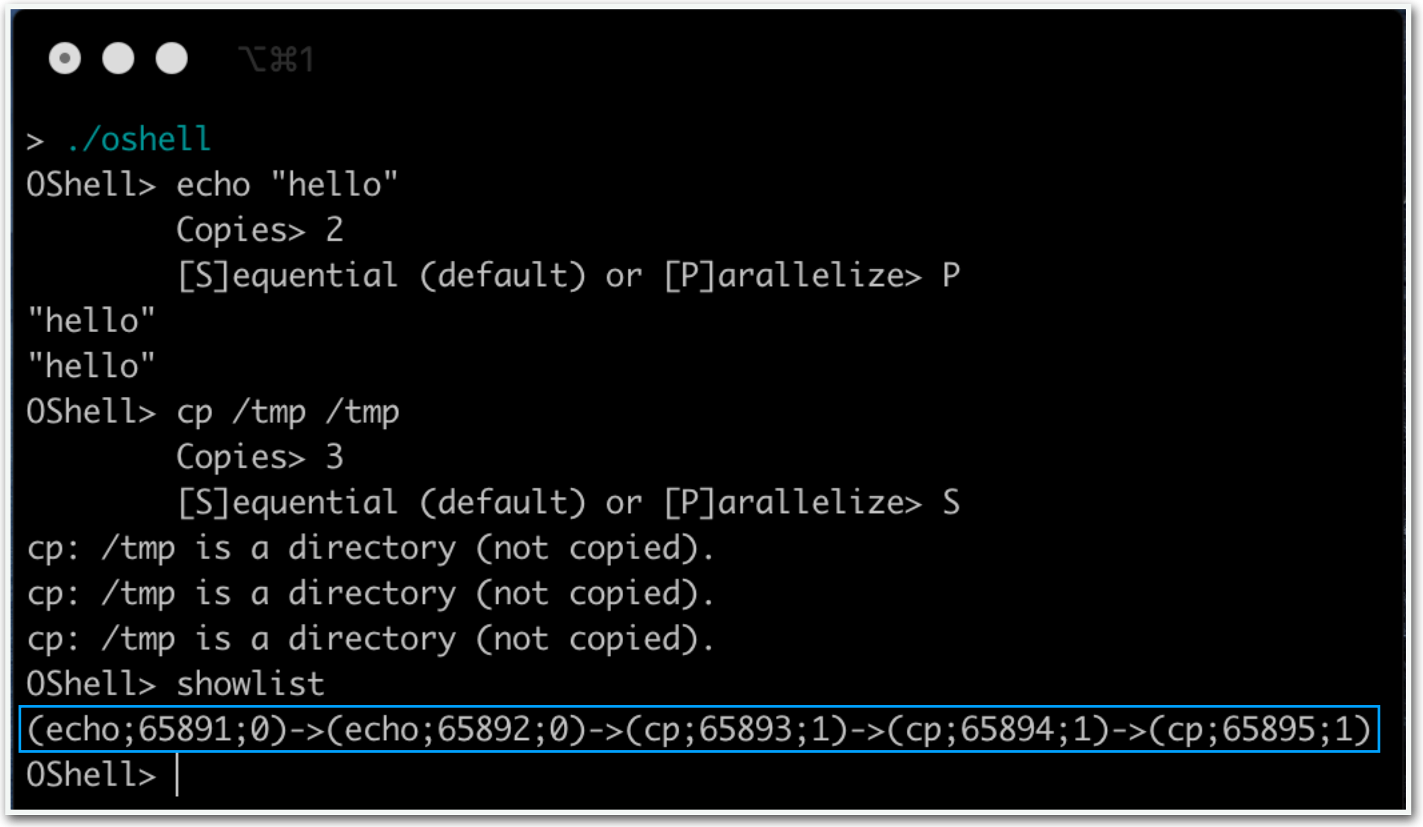
Parallel Execution:



Note: Use only `fork()` and `execvp()` to create a process.

The OShell:
3. List of process

Built-in command (see next slides)



```
⌥⌘1

> ./oshell
OShell> echo "hello"
        Copies> 2
        [S]equential (default) or [P]arallelize> P
"hello"
"hello"
OShell> cp /tmp /tmp
        Copies> 3
        [S]equential (default) or [P]arallelize> S
cp: /tmp is a directory (not copied).
cp: /tmp is a directory (not copied).
cp: /tmp is a directory (not copied).
OShell> showlist
(echo;65891;0)->(echo;65892;0)->(cp;65893;1)->(cp;65894;1)->(cp;65895;1)
OShell>
```

Use **dynamic** allocation to manage memory

You must save the list of processes that have been executed during a session.
Format: `(name1;pid1;exitCode1) –> … –> (nameN;pidN;exitCodeN)`

The OShell: 4. Built-in Commands (1)

For specific commands, you must handle on your own:

1. The `cd` command: to change directory;
2. The `exit` command: quite explicit;
3. The `showlist` command: to display the content of the process list on stdout;
4. The `memdump` command: to save the content of the process list into a binary file;
5. The `loadmem` command: to load the content of the binary file into the process list.

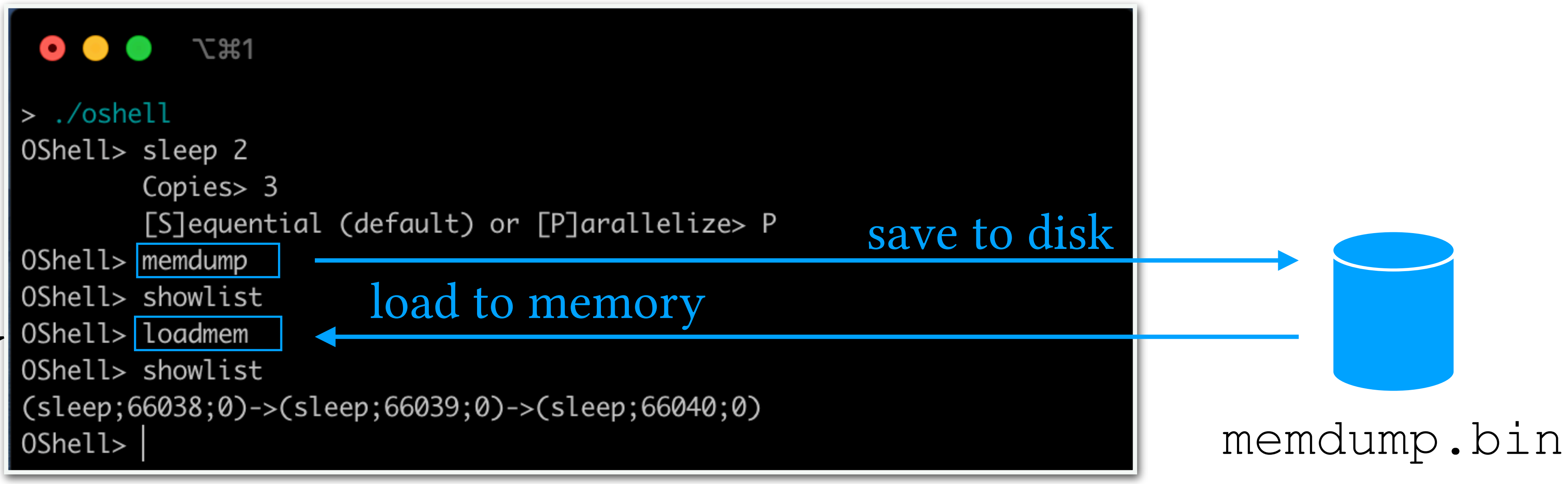
More info in next session

Note: Built-in commands are executed only once.

The OShell: 4. Built-in Commands (2)

Save the content of the list into a binary file. Then the list is freed.

Load data from binary file into the list in main memory.



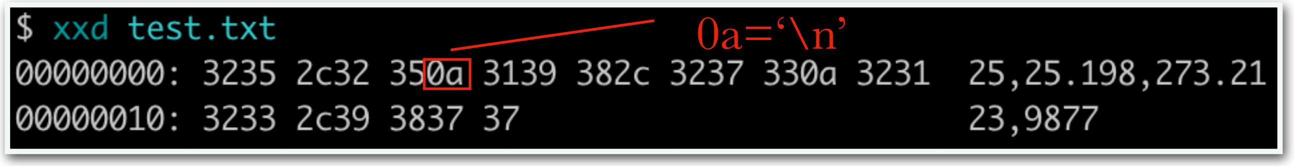
```
> ./oshell
OShell> sleep 2
        Copies> 3
        [S]equential (default) or [P]arallelize> P
OShell> memdump
OShell> showlist
OShell> loadmem
OShell> showlist
(sleep;66038;0)->(sleep;66039;0)->(sleep;66040;0)
OShell> |
```

THE OSHELL: BINARY FILES VS TEXT FILES

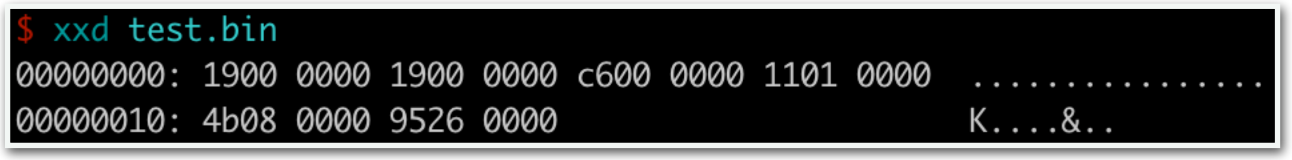
```c
struct Coord{
    int x; //4bytes
    int y; //4bytes
};

struct Coord c1 = {25,25};
struct Coord c2 = {198,273};
struct Coord c3 = {2123,9877};
```

Note: Architecture and compiler dependent



```
$ xxd test.txt
00000000: 3235 2c32 350a 3139 382c 3237 330a 3231  25,25.198,273.21
00000010: 3233 2c39 3837 37                        23,9877
```



```
$ xxd test.bin
00000000: 1900 0000 1900 0000 c600 0000 1101 0000  ................
00000010: 4b08 0000 9526 0000                      K....&..
```

Text files: store data using text representation (e.g., ASCII).

Binary files: store data using the same binary representation as the main memory.

https://www.scadacore.com/tools/programming-calculators/online-hex-converter/

The OShell

Demonstration

The OShell: Requirements

Others:

- Group of **two** that you will **keep** the whole semester
- Submit a tar file on the submission platform

Don't forget: We want clean code, without error, warning and memory leak (use `valgrind`).

Don't forget too: We detect **plagiarism** so don't try…

Plagiarism = **0 for the course!**

The OShell: Requirements

Deadline: 11th March 2020

Happy Coding!