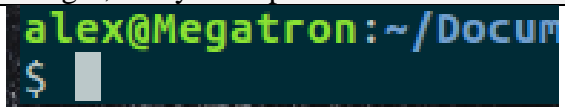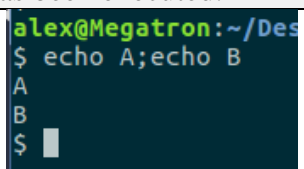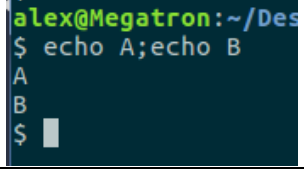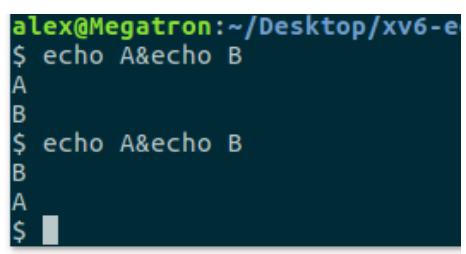Alexander M. Nicoara (A20301259)
CS450: Operating Systems
Programming Assignment 1
Professor Leung

Introduction

The objective of this lab was to implement 2 new execution operators: ";" for sequential command execution, and "&" for parallel command execution.

Test Cases

Here are the test cases for my modified xv6 shell:

Input	Expected Output	My Shell's Output	Status
Requirement 1			
Tests to see if the shell prompts the user for input by displaying the $ symbol with the text cursor proceeding it, ready for input.			
None	$ □		Pass
Requirement 2			
Tests to see if processes are executed in order when separated by **;**, and that a prompt appears after the last command has been executed.			
$ echo A;echo B	A B		Pass
$ echo A;echo B	A prompt ($ □) appears after the last command's execution.		Pass
Requirement 3			
Tests to see if processes are executed in parallel when separated by **&**, and that a prompt appears after the last command has been executed.			
$ echo A&echo B	A B B or A The output is not *deterministic*, meaning we could get either result.		Pass

$ echo A&echo B	A prompt ($) appears after the last command's execution.	`alex@Megatron:~/Desktop/xv6-e`
		`$ echo A&echo B`
		`A`
		`B`
		`$ echo A&echo B`
		`B`
		`A`
		`$ `
			Pass

Requirement 4 (Bonus Credit)

Tests to see if commands executed are real commands supported by the xv6 and that they take arguments.

$ echo A	A	`alex@Megatron:~/Desktop/xv6-public$`
		`$ echo A`
		`A`
		`$`
			Pass

$ ls	Lists current directory	`alex@Megatron:~/Desktop/xv6-public$ ./a.ou`
		`$ ls`
		`a.out      exec.c      kalloc.c   mem`
		`asm.h      fcntl.h     kbd.c      mem`
		`bio.c      file.c      kbd.h      _mk`
		`bootasm.s  file.h      kernel.ld  mk`
			Pass

$ spaghettisauce	Not a valid command	`alex@Megatron:~/Desktop/xv6-publi`
		`$ spaghettisauce`
		`exec failed: invalid xv6 command`
			Pass

$ abcdefg12345	Not a valid command	`$ abcdefg12345` `exec failed: invalid xv6 command`	Pass

Requirement 5

Tests to see if 3 or more commands can be run at a time, in a sequence, or in some combination of both.

$ echo A&echo B;echo C	A B
	B or A
	C C
	Result is not *deterministic*, we could get either one of these.
		`alex@Megatron:~/Desktop/xv6`
		`$ echo A&echo B;echo C`
		`B`
		`A`
		`C`
		`$ echo A&echo B;echo C`
		`A`
		`B`
		`C`
		`$`
			Pass

$ echo A;echo B& echo C	A A
	B or C
	C B
	Result is not *deterministic*, we could get either one of these.
		`alex@Megatron:~/Desktop/xv`
		`$ echo A;echo B&echo C`
		`A`
		`B`
		`C`
		`$ echo A;echo B&echo C`
		`A`
		`C`
		`B`
			Pass

$ echo A;echo B;echo C	A
	B
	C
		`alex@Megatron:~/Desktop/xv6-p`
		`$ echo A;echo B;echo C`
		`A`
		`B`
		`C`
		`$`
			Pass

| $ echo A&echo B&echo C | A  B  C  C<br>B  A  A  B<br>C  C  B  A<br>   A     B<br>   C     C<br>   B     A<br>Any one of the results above is possible since the execution result is not *deterministic*. | ```
$ echo A&echo B&echo C
B
A
C
$ echo A&echo B&echo C
A
B
C
$ echo A&echo B&echo C
A
B
C
$ echo A&echo B&echo C
C
B
A
$ echo A&echo B&echo C
B
C
A
$ echo A&echo B&echo C
C
B
A
$ echo A&echo B&echo C
A
C
B
$ echo A&echo B&echo C
A
B
C
$ echo A&echo B&echo C
C
B
A
$ echo A&echo B&echo C
C
A
B
$ ^C
``` | Pass |
| **Requirement 6** | | | | |
| Tests to see if process terminates after ; (since there is no other command after) and that it can't run a process ending with &. | | | | |
| $ echo abc; | abc | ```
$ echo abc;
abc
$
``` | Pass |
| $ echo B& | Failed to exec, can't terminate with & | ```
alex@Megatron:~/Desktop/xv6-edit$ ./a.out
$ echo A&
failed to exec: can't terminate with a &
$
``` | Pass |

Validation of my test data

I believe my test data is sufficient because it lists cases/output that verify and meet all the requirements of the assignment. In some instances of my case tests (like the 3 or more commands requirement), I added more cases to test most or all combinations (with the operators) to show that it can produce the expected output if given 3 random commands with random operators. In precise and small requirements, I used less cases to just show that the mechanism works. I wanted to avoid duplicating test cases they might already be shown in the document.