Program 1 (System Calls and Shell)

Reza Naeemi

Purpose

This assignment intends (1) to familiarize you with Linux programming using several system calls such as *fork*, *execlp*, *wait*, *pipe*, *dup2*, and *close*, and (2) to help you understand that, from the kernel's view point, the *shell* is simply viewed as an application program that uses system calls to spawn and to terminate other user programs. You will also become familiar with the *ThreadOS* operating system simulator in part 2 of this assignment.

Documentation

processes

fork() system call is used to create a new process, which becomes the child of the caller process. It takes no parameters and return integer values, which represent if it was successful or not. The following are different values returned by the fork() function:

- Negative integer: creation of a child process was unsuccessful
- Zero: Returned to the newly created child process.
- Positive integer: Returned to parent or caller, the integer represents

How does the shell execute a user command? The mechanism follows the steps given below:

1. The shell locates the executable file whose name is specified in the first string given from the keyboard input.
2. It creates a child process by duplicating itself.
3. The duplicated shell overloads its process image with the executable file.
4. The overloaded process receives all the remaining strings given from the keyboard input, and starts the command execution.

The Shell can receive two or more commands at a time from the keyboard input. The symbols ';' and '&' are used as a delimiter specifying the end of each single command. If a command is delimited by ';', the shell spawns a new process to execute this command, then it waits for the process to be terminated before continuing. If the command is delimited by '&', the shell spawns a new process to execute this command, and thereafter continues to interpret the next command without waiting for the completion of the current command.

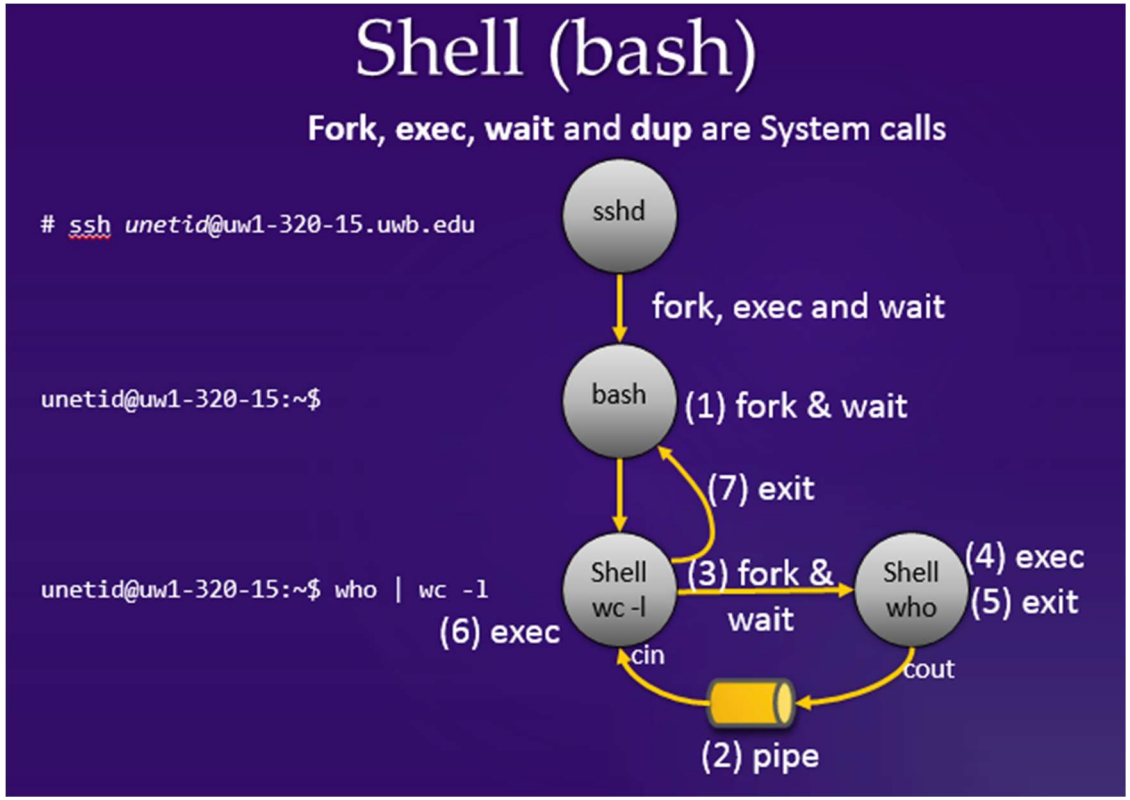
I/O redirection and pipeline

One of the shell's interesting features is In and Out (I/O) redirection. This allows the redirection of the standard input or output to a file. Another convenience feature is pipeline, which connects one command's standard output to the standard input of another command. This is very useful when trying to filter down outputs and get the exact results wanted.

The following are the functions to accomplish pipeline redirection and creating child processes and so on:

- pid_t fork(void); creates a child process that differs from the parent process only in terms of their process IDs.
- int execlp(const char* file, const char* args, … , (char*)0); replaces the current process image with a new process image that will be loaded from file. The first argument arg must be the same as file.
- int pipe(int filedes[2]); creates a pair of file descriptors (which point to a pipe structure), and places them in the array pointed to by filedes. filedes[0] is for reading data from the pipe, filedes[1] is for writing data to the pipe.
- int dup2(int oldfd, int newfd); creates in newfd a copy of the file descriptor oldfd.
- pid_t wait(int* status); waits for process termination
- int close(int fd); closes a file descriptor

In our case, we imitated how the shell performs "ps –A | grep argv[1] | wc –l". In other words, our parent process spawned a child that spawned a grand-child that then spawned a great-grand-child. We use dup2 to redirect stdout to the std in of our pipe and into the next command as an stdin.

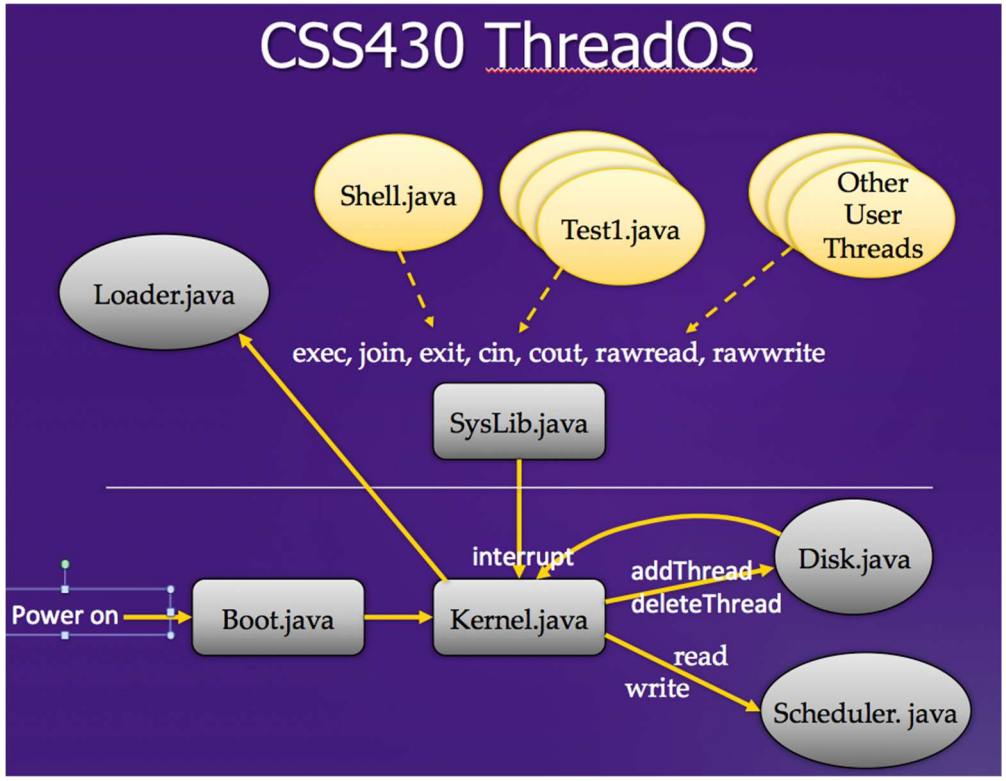


ThreadOS and Shell (Java Thread)

ThreadOS is an operating system simulator that has been designed in Java for educational institutions to use in Computer Science and Software Engineering courses.

Thread OS loads Java programs that have been derived from the Thread class, manages them as user processes, and provides them with some basic operating system services. Those services include thread spawn, thread termination, disk operation, and even standard input/output. Thread OS

receives all service requests as a form of simulated interrupt from each user thread, handles them, and returns a status value to the interrupting thread.



ThreadOS Structure and Components

- Boot (Boot.java) – invokes a BOOT system call to have Kernel initialize its internal data power on Disk, start the scheduler thread, and finally spawn the Loader thread.

- Kernel (Kernel.java) – receives an interrupt, services it if possible, otherwise forwards its request to the scheduler on Disk, and returns a completion status.

- Disk (Disk.java) – simulates a slow disk device composed of 1000 blocks, each containing 512 bytes. Those blocks are divided into 10 tracks, each of which thus includes 100 blocks. The disk has three commands: read, write, and sync.

- Scheduler (Scheduler.java, TCB.java) – receives a Thread object that Kernel instantiated upon receiving an EXEC system call, allocates a new TCB (Thread Control Block) to this thread, enqueues the TCB into its ready queue, and schedules its execution in a round robin fashion.

- SysLib (SysLib.java) – is a utility that provides user threads with a convenient style of system calls and converts them into corresponding interrupts passed to the Kernel.

To start ThreadOS, simply type: *java Boot*

Boot initializes Kernel data, powers on Disk, and starts Scheduler. Its finally launches Loader that then carries out one of the following commands:

- ***?*** – prints out its usage
- ***l Shell*** – starts the Shell program thread as an independent user thread and waits for its termination. Shell is our User Program in this case.
- ***q*** – synchronizes disk data and terminates ThreadOS

User Programs

A user program must be a subclass of the Java Thread class. Java threads are execution entities concurrently running in a Java application. They maintain their own stacks and program counter but share static variables in their application.

System Class

ThreadOS Kernel receives requests from each user thread as interrupts to it. Since this interrupt method is not an elegant form to a user program, ThreadOS provides a user program with its system library, called SysLib that includes several important system-call functions as shown below. (Unless otherwise mentioned, each of these functions return 0 on success or -1 on error.)

1. SysLib.exe(String args[]) loads the class specified in args[0], instantiates its object, simply passes the following element of the String array, and starts it as a child thread. It returns a child thread ID on success, otherwise -1.

2. SysLib.join() waits for termination of one of child threads. It returns the ID of the child thread that has woken up calling thread. If it fails, it returns -1.

3. SysLib.exit() terminates the calling thread and wakes up its parent thread if this parent is waiting on join().

4. SysLib.cin(StringBuffer s) reads keyboard input to the StringBuffer s.

5. SysLib.cout(String s) prints out the String s to the standard output. “\n” is supported.

6. SysLib.cerr(String s) prints out the String s to the standard error. “\n” is supported.

7. SysLib.sync() writes back all memory data to disk.

In addition to those system calls, the system library includes several utility functions. One of them is:

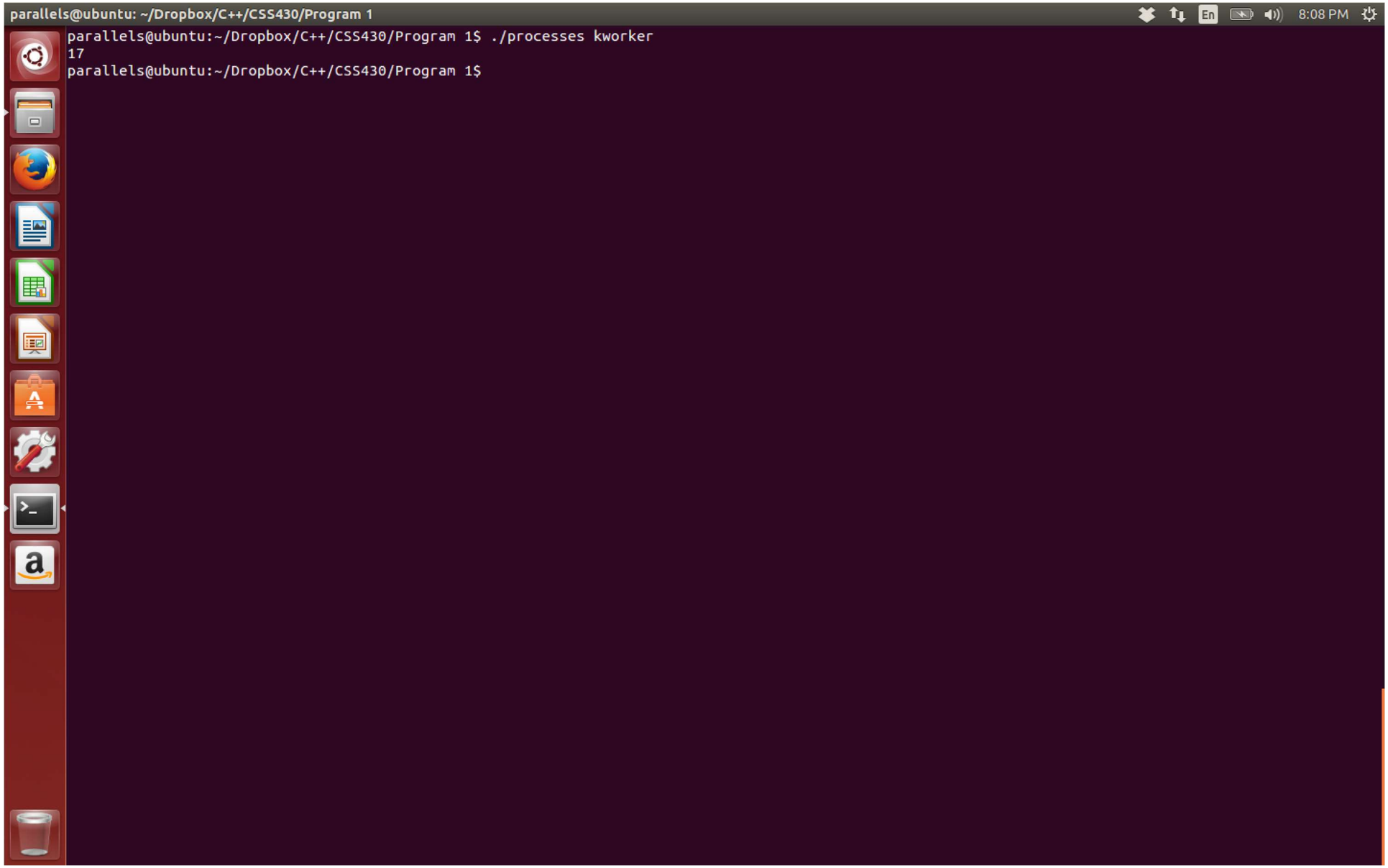
1. Public static String[] SysLib.stringToArgs(String s) converts a space-delimited string into a String array in that each space-delimited String is stored into a different array element. This call returns such a String array.

Execution Console Output (Both processes and Shell in ThreadOS)

I used my processes executable and passed in kworker into it. Kworker is the process I wanted the count for, and as you can see it requested 17. This was clearly the count that wc returned at the end, after all the Strings were passed through our pipelines.

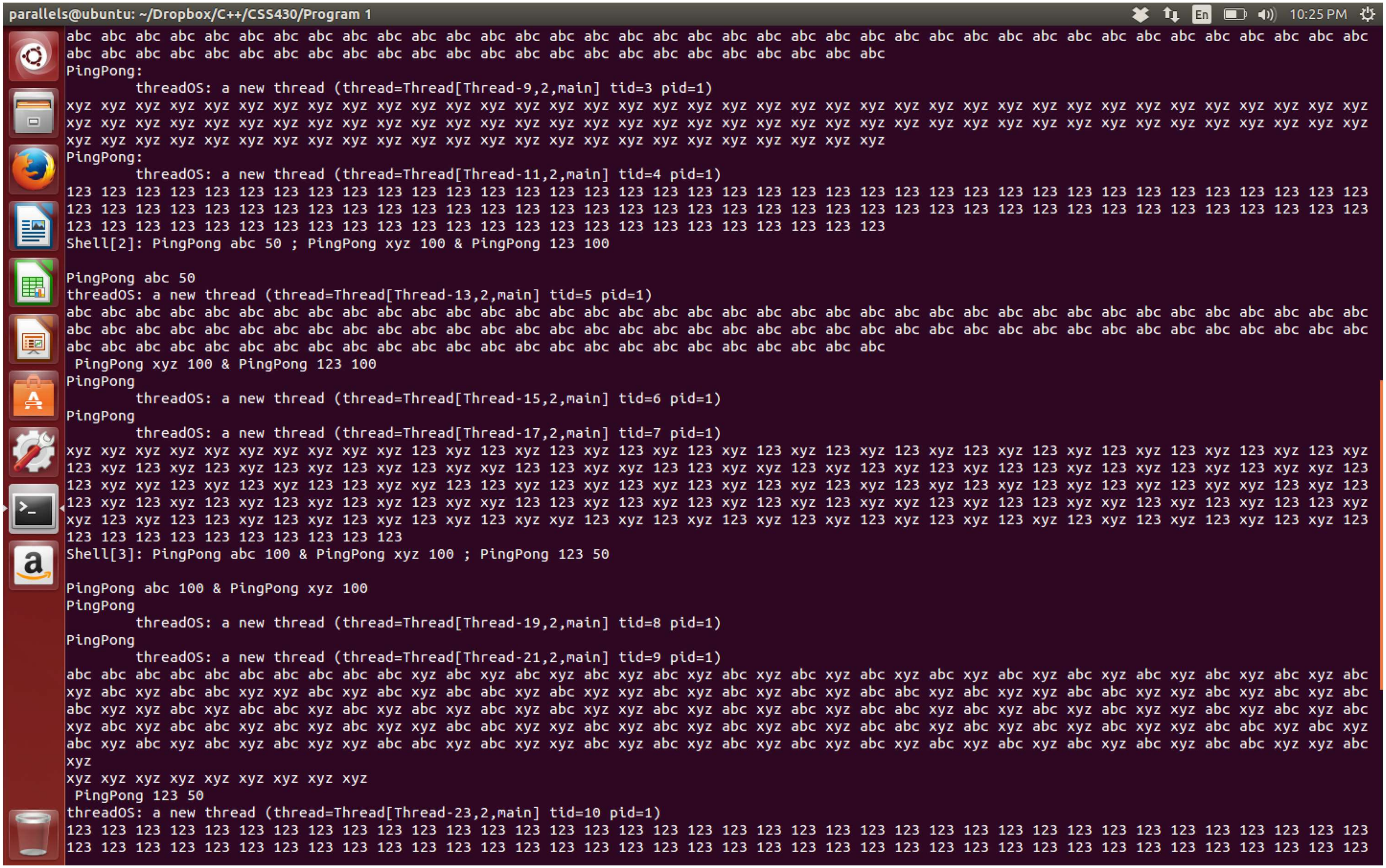
To run our Shell Thread in ThreadOS, we did the following:

1. *java Boot*, to launch and get Thread OS ready.
2. *l Shell*, to start our Shell thread.
3. Finally, I ran the following tests:
    a. Shell[1]% *PingPong abc 100 ; PingPong xyz 50 ; PingPong 123 100*
    b. Shell[2]% *PingPong abc 50 ; PingPong xyz 100 & PingPong 123 100*
    c. Shell[3]% *PingPong abc 100 & PingPong xyz 100 ; PingPong 123 50*
    d. Shell[4]% *PingPong abc 50 & PingPong xyz 50 & PingPong 123 100*


parallels@ubuntu: ~/Dropbox/C++/CSS430/Program 1
8:08 PM
parallels@ubuntu:~/Dropbox/C++/CSS430/Program 1$ ./processes kworker
17
parallels@ubuntu:~/Dropbox/C++/CSS430/Program 1$

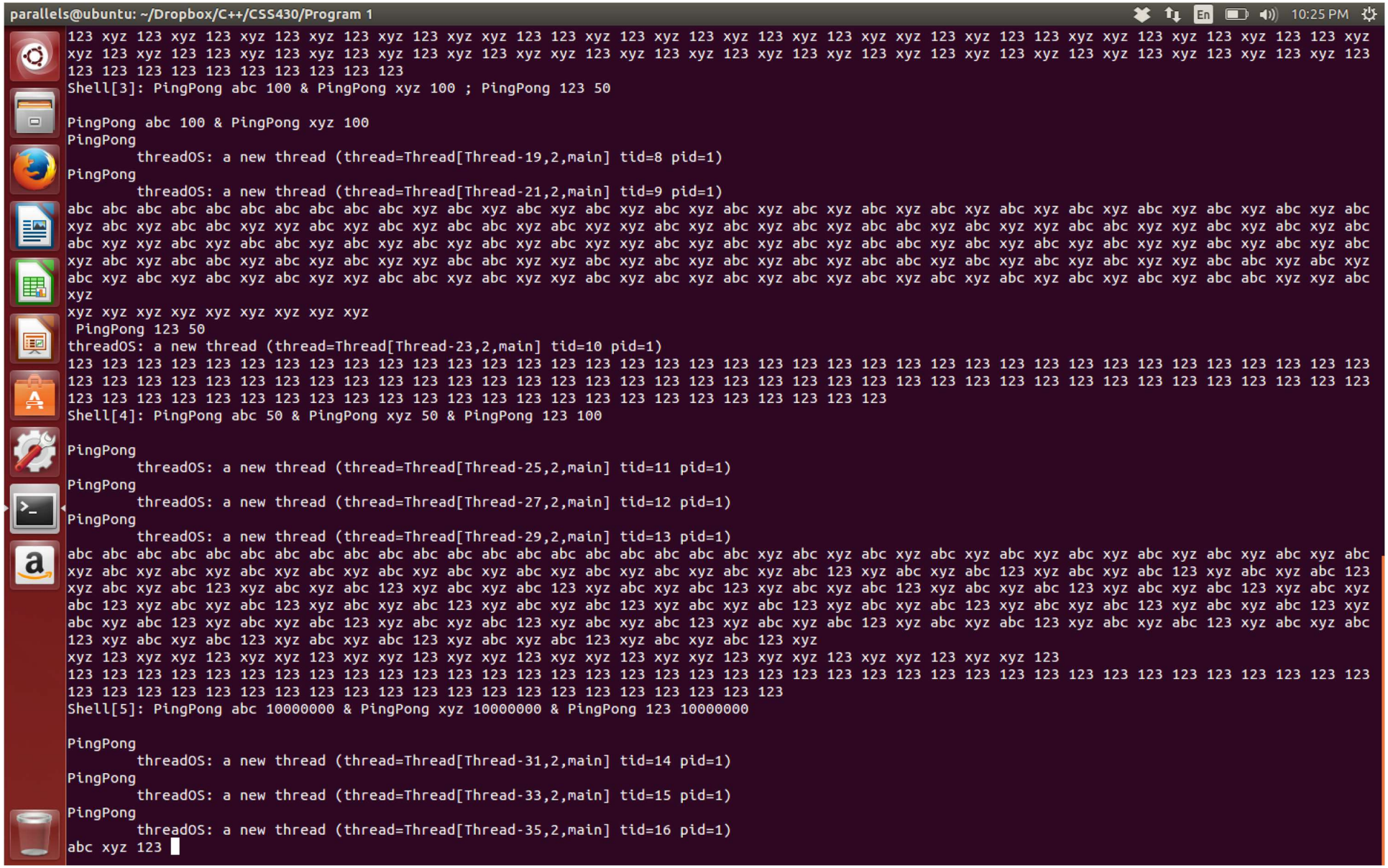
parallels@ubuntu: ~/Dropbox/C++/CSS430/Program 1 10:25 PM

```
parallels@ubuntu:~/Dropbox/C++/CSS430/Program 1$ javac Shell.java
parallels@ubuntu:~/Dropbox/C++/CSS430/Program 1$ java Boot
threadOS ver 1.0:
Type ? for help
threadOS: a new thread (thread=Thread[Thread-3,2,main] tid=0 pid=-1)
-->l Shell
l Shell
Welcome to my Shell!threadOS: a new thread (thread=Thread[Thread-5,2,main] tid=1 pid=0)
Shell[1]: exit
Exiting...
Superblock synchronized
-->q
q
Superblock synchronized
parallels@ubuntu:~/Dropbox/C++/CSS430/Program 1$ rm Shell.class
parallels@ubuntu:~/Dropbox/C++/CSS430/Program 1$ nano Shell.java
parallels@ubuntu:~/Dropbox/C++/CSS430/Program 1$ javac Shell.java
parallels@ubuntu:~/Dropbox/C++/CSS430/Program 1$ java Boot
threadOS ver 1.0:
Type ? for help
threadOS: a new thread (thread=Thread[Thread-3,2,main] tid=0 pid=-1)
-->l Shell
l Shell
Welcome to my Shell!
threadOS: a new thread (thread=Thread[Thread-5,2,main] tid=1 pid=0)
Shell[1]: PingPong abc 100 ; PingPong xyz 50 ; PingPong 123 100

PingPong:
        threadOS: a new thread (thread=Thread[Thread-7,2,main] tid=2 pid=1)
abc abc abc abc abc abc abc abc abc abc abc abc abc abc abc abc abc abc abc abc abc abc abc abc abc abc abc abc abc abc abc abc abc abc abc abc abc abc
abc abc abc abc abc abc abc abc abc abc abc abc abc abc abc abc abc abc abc abc abc abc abc abc abc abc abc abc abc abc abc abc abc abc abc abc abc abc
abc abc abc abc abc abc abc abc abc abc abc abc abc abc abc abc abc abc abc abc abc abc abc abc
PingPong:
        threadOS: a new thread (thread=Thread[Thread-9,2,main] tid=3 pid=1)
xyz xyz xyz xyz xyz xyz xyz xyz xyz xyz xyz xyz xyz xyz xyz xyz xyz xyz xyz xyz xyz xyz xyz xyz xyz xyz xyz xyz xyz xyz xyz xyz xyz xyz xyz xyz xyz xyz
xyz xyz xyz xyz xyz xyz xyz xyz xyz xyz xyz xyz xyz xyz xyz xyz xyz xyz xyz xyz xyz xyz xyz xyz xyz xyz xyz xyz xyz xyz xyz xyz xyz xyz xyz xyz xyz xyz
xyz xyz xyz xyz xyz xyz xyz xyz xyz xyz xyz xyz xyz xyz xyz xyz xyz xyz xyz xyz xyz xyz xyz xyz
PingPong:
        threadOS: a new thread (thread=Thread[Thread-11,2,main] tid=4 pid=1)
123 123 123 123 123 123 123 123 123 123 123 123 123 123 123 123 123 123 123 123 123 123 123 123 123 123 123 123 123 123 123 123 123 123 123 123 123 123
123 123 123 123 123 123 123 123 123 123 123 123 123 123 123 123 123 123 123 123 123 123 123 123 123 123 123 123 123 123 123 123 123 123 123 123 123 123
123 123 123 123 123 123 123 123 123 123 123 123 123 123 123 123 123 123 123 123 123 123 123 123
Shell[2]: PingPong abc 50 ; PingPong xyz 100 & PingPong 123 100

PingPong abc 50
threadOS: a new thread (thread=Thread[Thread-13,2,main] tid=5 pid=1)
abc abc abc abc abc abc abc abc abc abc abc abc abc abc abc abc abc abc abc abc abc abc abc abc abc abc abc abc abc abc abc abc abc abc abc abc abc abc
abc abc abc abc abc abc abc abc abc abc abc abc abc abc abc abc abc abc abc abc abc abc abc abc abc abc abc abc abc abc abc abc abc abc abc abc abc abc
```


parallels@ubuntu: ~/Dropbox/C++/CSS430/Program 1
En
10:25 PM
abc abc abc abc abc abc abc abc abc abc abc abc abc abc abc abc abc abc abc abc abc abc abc abc abc abc abc abc abc abc abc abc abc abc abc abc abc abc
abc abc abc abc abc abc abc abc abc abc abc abc abc abc abc abc abc abc abc abc abc abc abc abc
PingPong:
threadOS: a new thread (thread=Thread[Thread-9,2,main] tid=3 pid=1)
xyz xyz xyz xyz xyz xyz xyz xyz xyz xyz xyz xyz xyz xyz xyz xyz xyz xyz xyz xyz xyz xyz xyz xyz xyz xyz xyz xyz xyz xyz xyz xyz xyz xyz xyz xyz xyz xyz
xyz xyz xyz xyz xyz xyz xyz xyz xyz xyz xyz xyz xyz xyz xyz xyz xyz xyz xyz xyz xyz xyz xyz xyz xyz xyz xyz xyz xyz xyz xyz xyz xyz xyz xyz xyz xyz xyz
xyz xyz xyz xyz xyz xyz xyz xyz xyz xyz xyz xyz xyz xyz xyz xyz xyz xyz xyz xyz xyz xyz xyz xyz
PingPong:
threadOS: a new thread (thread=Thread[Thread-11,2,main] tid=4 pid=1)
123 123 123 123 123 123 123 123 123 123 123 123 123 123 123 123 123 123 123 123 123 123 123 123 123 123 123 123 123 123 123 123 123 123 123 123 123 123
123 123 123 123 123 123 123 123 123 123 123 123 123 123 123 123 123 123 123 123 123 123 123 123 123 123 123 123 123 123 123 123 123 123 123 123 123 123
123 123 123 123 123 123 123 123 123 123 123 123 123 123 123 123 123 123 123 123 123 123 123 123
Shell[2]: PingPong abc 50 ; PingPong xyz 100 & PingPong 123 100

PingPong abc 50
threadOS: a new thread (thread=Thread[Thread-13,2,main] tid=5 pid=1)
abc abc abc abc abc abc abc abc abc abc abc abc abc abc abc abc abc abc abc abc abc abc abc abc abc abc abc abc abc abc abc abc abc abc abc abc abc abc
abc abc abc abc abc abc abc abc abc abc abc abc abc abc abc abc abc abc abc abc abc abc abc abc abc abc abc abc abc abc abc abc abc abc abc abc abc abc
abc abc abc abc abc abc abc abc abc abc abc abc abc abc abc abc abc abc abc abc abc abc abc abc
PingPong xyz 100 & PingPong 123 100
PingPong
threadOS: a new thread (thread=Thread[Thread-15,2,main] tid=6 pid=1)
PingPong
threadOS: a new thread (thread=Thread[Thread-17,2,main] tid=7 pid=1)
xyz xyz xyz xyz xyz xyz xyz xyz xyz xyz 123 xyz 123 xyz 123 xyz 123 xyz 123 xyz 123 xyz 123 xyz 123 xyz 123 xyz 123 xyz 123 xyz 123 xyz 123 xyz 123 xyz
123 xyz 123 xyz 123 xyz 123 xyz 123 xyz 123 xyz xyz 123 123 xyz xyz 123 123 xyz xyz 123 xyz 123 xyz 123 xyz 123 123 xyz 123 xyz 123 xyz 123 xyz xyz 123
123 xyz xyz 123 xyz 123 xyz 123 123 xyz xyz 123 123 xyz 123 xyz 123 xyz 123 xyz 123 xyz 123 xyz 123 xyz 123 xyz 123 xyz 123 xyz 123 xyz xyz 123 xyz 123
123 xyz 123 xyz 123 xyz 123 xyz 123 xyz 123 xyz xyz 123 123 xyz 123 xyz 123 xyz 123 xyz 123 xyz xyz 123 xyz 123 123 xyz xyz 123 xyz 123 xyz 123 123 xyz
xyz 123 xyz 123 123 xyz 123 xyz 123 xyz 123 xyz 123 xyz xyz 123 xyz 123 xyz 123 xyz 123 xyz 123 xyz 123 xyz 123 xyz 123 xyz 123 xyz 123 xyz 123 xyz 123
123 123 123 123 123 123 123 123 123 123
Shell[3]: PingPong abc 100 & PingPong xyz 100 ; PingPong 123 50

PingPong abc 100 & PingPong xyz 100
PingPong
threadOS: a new thread (thread=Thread[Thread-19,2,main] tid=8 pid=1)
PingPong
threadOS: a new thread (thread=Thread[Thread-21,2,main] tid=9 pid=1)
abc abc abc abc abc abc abc abc abc abc xyz abc xyz abc xyz abc xyz abc xyz abc xyz abc xyz abc xyz abc xyz abc xyz abc xyz abc xyz abc xyz abc xyz abc
xyz abc xyz abc abc xyz xyz abc xyz abc xyz abc abc xyz abc xyz xyz abc xyz abc xyz abc xyz abc abc xyz abc xyz xyz abc abc xyz xyz abc xyz abc xyz abc
abc xyz xyz abc xyz abc abc xyz abc xyz abc xyz abc xyz abc xyz xyz abc xyz abc xyz abc xyz abc abc xyz abc xyz abc xyz abc xyz xyz abc xyz abc xyz abc
xyz abc xyz abc abc xyz abc xyz abc xyz xyz abc abc xyz xyz abc xyz abc xyz abc xyz abc xyz abc abc xyz abc xyz abc xyz abc xyz xyz abc abc xyz abc xyz
abc xyz abc xyz abc xyz abc xyz xyz abc abc xyz abc xyz xyz abc xyz abc xyz abc xyz abc xyz abc xyz abc xyz abc xyz abc xyz abc xyz abc abc xyz xyz abc
xyz
xyz xyz xyz xyz xyz xyz xyz xyz xyz
PingPong 123 50
threadOS: a new thread (thread=Thread[Thread-23,2,main] tid=10 pid=1)
123 123 123 123 123 123 123 123 123 123 123 123 123 123 123 123 123 123 123 123 123 123 123 123 123 123 123 123 123 123 123 123 123 123 123 123 123 123
123 123 123 123 123 123 123 123 123 123 123 123 123 123 123 123 123 123 123 123 123 123 123 123 123 123 123 123 123 123 123 123 123 123 123 123 123 123


parallels@ubuntu: ~/Dropbox/C++/CSS430/Program 1
10:25 PM
123 xyz 123 xyz 123 xyz 123 xyz 123 xyz 123 xyz xyz 123 123 xyz 123 xyz 123 xyz 123 xyz 123 xyz xyz 123 xyz 123 123 xyz xyz 123 xyz 123 xyz 123 123 xyz
xyz 123 xyz 123 123 xyz 123 xyz 123 xyz 123 xyz 123 xyz xyz 123 xyz 123 xyz 123 xyz 123 xyz 123 xyz 123 xyz 123 xyz 123 xyz 123 xyz 123 xyz 123 xyz 123 xyz 123
123 123 123 123 123 123 123 123 123 123
Shell[3]: PingPong abc 100 & PingPong xyz 100 ; PingPong 123 50
PingPong abc 100 & PingPong xyz 100
PingPong
threadOS: a new thread (thread=Thread[Thread-19,2,main] tid=8 pid=1)
PingPong
threadOS: a new thread (thread=Thread[Thread-21,2,main] tid=9 pid=1)
abc abc abc abc abc abc abc abc abc abc xyz abc xyz abc xyz abc xyz abc xyz abc xyz abc xyz abc xyz abc xyz abc xyz abc xyz abc xyz abc
xyz abc xyz abc abc xyz xyz abc xyz abc xyz abc abc xyz abc xyz xyz abc xyz abc xyz abc xyz abc abc xyz abc xyz xyz abc abc xyz xyz abc xyz abc xyz abc
abc xyz xyz abc xyz abc abc xyz abc xyz abc xyz abc xyz abc xyz xyz abc xyz abc xyz abc xyz abc abc xyz abc xyz abc xyz abc xyz xyz abc xyz abc xyz abc
xyz abc xyz abc abc xyz abc xyz abc xyz xyz abc abc xyz xyz abc xyz abc xyz abc xyz abc xyz abc abc xyz abc xyz abc xyz abc xyz xyz abc abc xyz abc xyz
abc xyz abc xyz abc xyz abc xyz xyz abc abc xyz abc xyz xyz abc xyz abc xyz abc xyz abc xyz abc xyz abc xyz abc xyz abc xyz abc abc xyz xyz abc
xyz
xyz xyz xyz xyz xyz xyz xyz xyz xyz
PingPong 123 50
threadOS: a new thread (thread=Thread[Thread-23,2,main] tid=10 pid=1)
123 123 123 123 123 123 123 123 123 123 123 123 123 123 123 123 123 123 123 123 123 123 123 123 123 123 123 123 123 123 123 123 123 123 123 123 123
123 123 123 123 123 123 123 123 123 123 123 123 123 123 123 123 123 123 123 123 123 123 123 123 123 123 123 123 123 123 123 123 123 123 123 123 123
123 123 123 123 123 123 123 123 123 123 123 123 123 123 123 123 123 123 123 123 123 123 123 123 123 123 123 123 123 123 123 123 123 123 123 123 123
Shell[4]: PingPong abc 50 & PingPong xyz 50 & PingPong 123 100
PingPong
threadOS: a new thread (thread=Thread[Thread-25,2,main] tid=11 pid=1)
PingPong
threadOS: a new thread (thread=Thread[Thread-27,2,main] tid=12 pid=1)
PingPong
threadOS: a new thread (thread=Thread[Thread-29,2,main] tid=13 pid=1)
abc abc abc abc abc abc abc abc abc abc abc abc abc abc abc abc abc abc abc abc xyz abc xyz abc xyz abc xyz abc xyz abc xyz abc xyz abc xyz abc xyz abc
xyz abc xyz abc xyz abc xyz abc xyz abc xyz abc xyz abc xyz abc xyz abc xyz abc xyz abc xyz abc 123 xyz abc xyz abc 123 xyz abc xyz abc 123 xyz abc xyz abc
xyz abc xyz abc 123 xyz abc xyz abc 123 xyz abc xyz abc 123 xyz abc xyz abc 123 xyz abc xyz abc 123 xyz abc xyz abc 123 xyz abc xyz abc 123 xyz abc xyz
abc 123 xyz abc xyz abc 123 xyz abc xyz abc 123 xyz abc xyz abc 123 xyz abc xyz abc 123 xyz abc xyz abc 123 xyz abc xyz abc 123 xyz abc xyz abc 123 xyz
abc xyz abc 123 xyz abc xyz abc 123 xyz abc xyz abc 123 xyz abc xyz abc 123 xyz abc xyz abc 123 xyz abc xyz abc 123 xyz abc xyz abc 123 xyz abc xyz abc
123 xyz abc xyz abc 123 xyz abc xyz abc 123 xyz abc xyz abc 123 xyz abc xyz abc 123 xyz
xyz 123 xyz xyz 123 xyz xyz 123 xyz xyz 123 xyz xyz 123 xyz xyz 123 xyz xyz 123 xyz xyz 123 xyz xyz 123 xyz xyz 123 xyz xyz 123
123 123 123 123 123 123 123 123 123 123 123 123 123 123 123 123 123 123 123 123 123 123 123 123 123 123 123 123 123 123 123 123 123 123 123 123 123
123 123 123 123 123 123 123 123 123 123 123 123 123 123 123 123 123 123 123 123 123 123 123 123 123 123 123 123 123 123 123 123 123 123 123 123 123
Shell[5]: PingPong abc 10000000 & PingPong xyz 10000000 & PingPong 123 10000000
PingPong
threadOS: a new thread (thread=Thread[Thread-31,2,main] tid=14 pid=1)
PingPong
threadOS: a new thread (thread=Thread[Thread-33,2,main] tid=15 pid=1)
PingPong
threadOS: a new thread (thread=Thread[Thread-35,2,main] tid=16 pid=1)
abc xyz 123