If $73_x + 50_x = 103_x$ then what is base x?

Answer: 12

Using the definition of what it means to write a number in a given base, this means $(7x^1 + 3x^0) + (5x^1 + 0x^0) = 1x^2 + 0x^1 + 3x^0$. Simplifying, we have $12x + 3 = x^2 + 3$, so $12x = x^2$. Thus, either $x = 0$ or $x = 12$. By the definition we used earlier, number bases must be positive integers, so the answer is $x = 12$.

Suppose 18-bit instructions are copied to a CPU's instruction register, and suppose that each instruction dedicates the first 7 bits for the "opcode", with the remaining bits for specifying a memory location. What is the total number of possible memory locations that could be referenced by these instructions?

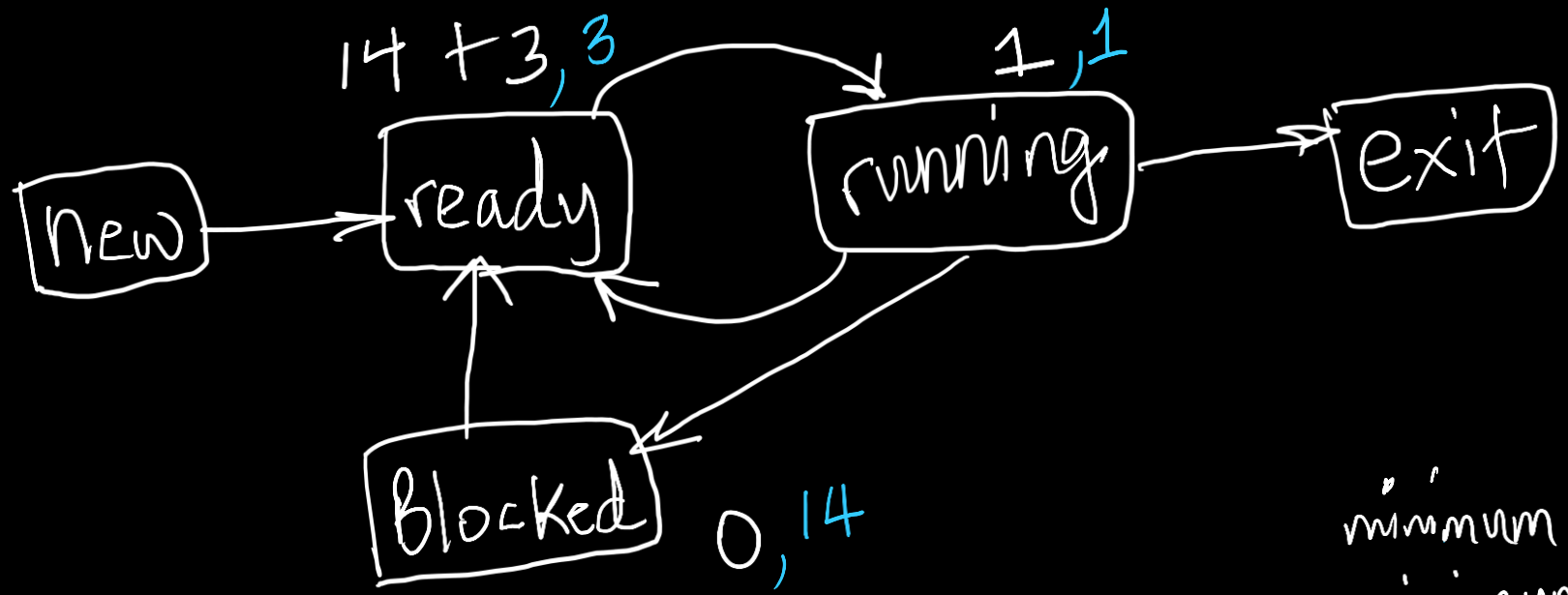
Answer: 2048

Since, in an instruction, 7 bits out of 18 bits are used for the "opcode", this means 18 - 7 = 11 bits are used for the address. So, there are "2 to the power 11" (i.e. 2048) possible memory locations that could be referenced by the instructions.

Assume the 5 state process model. Suppose 18 processes have been admitted from the "new" state and are all executing on a system with a single CPU, where 14 of these processes make I/O requests. Assume that at any given time, one of these processes is assigned to the CPU. Based only on this information, before any of these processes terminate, **what is the minimum number of processes that can be in the:**

1) ready state; 3

2) blocked state? 0



18 processes

14 make I/O req,

4 are CPU bound.

minimum in ready state = 3

minimum in blocked state = 0

[In this question bold text indicates Unix commands]

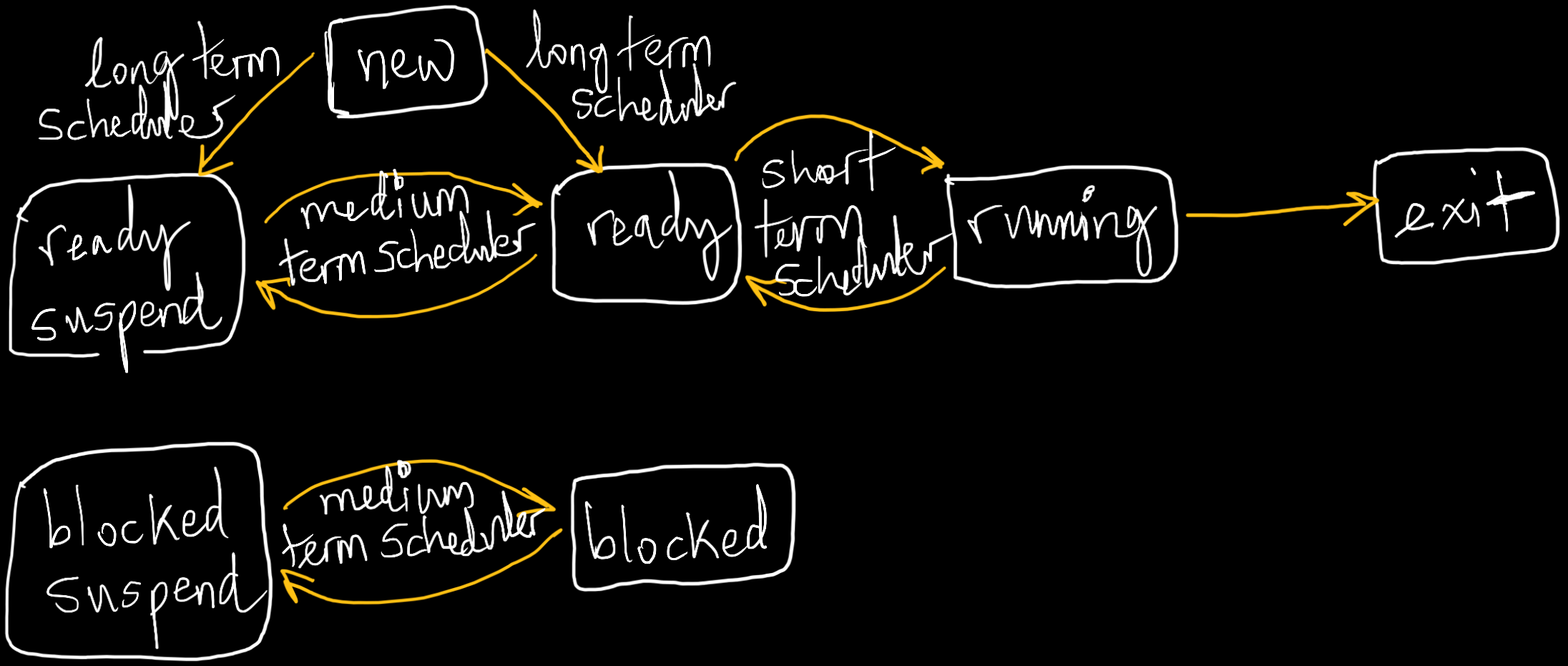
A process is an executing program. The Unix commands **man, echo** and **ls** are programs.
When the compound command **man man ; man echo ; echo $$; ls -al** is successfully executed in a Unix shell, how many processes are created?

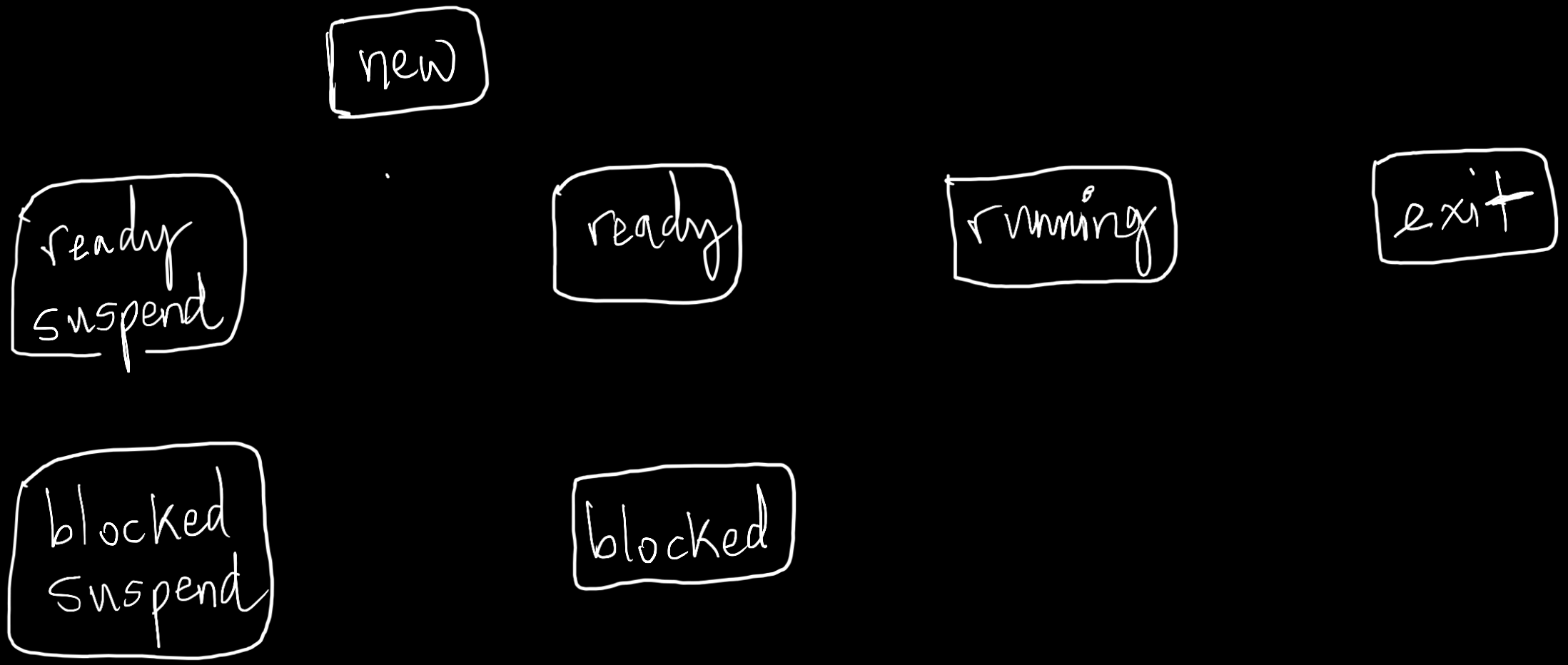
Answer: 4

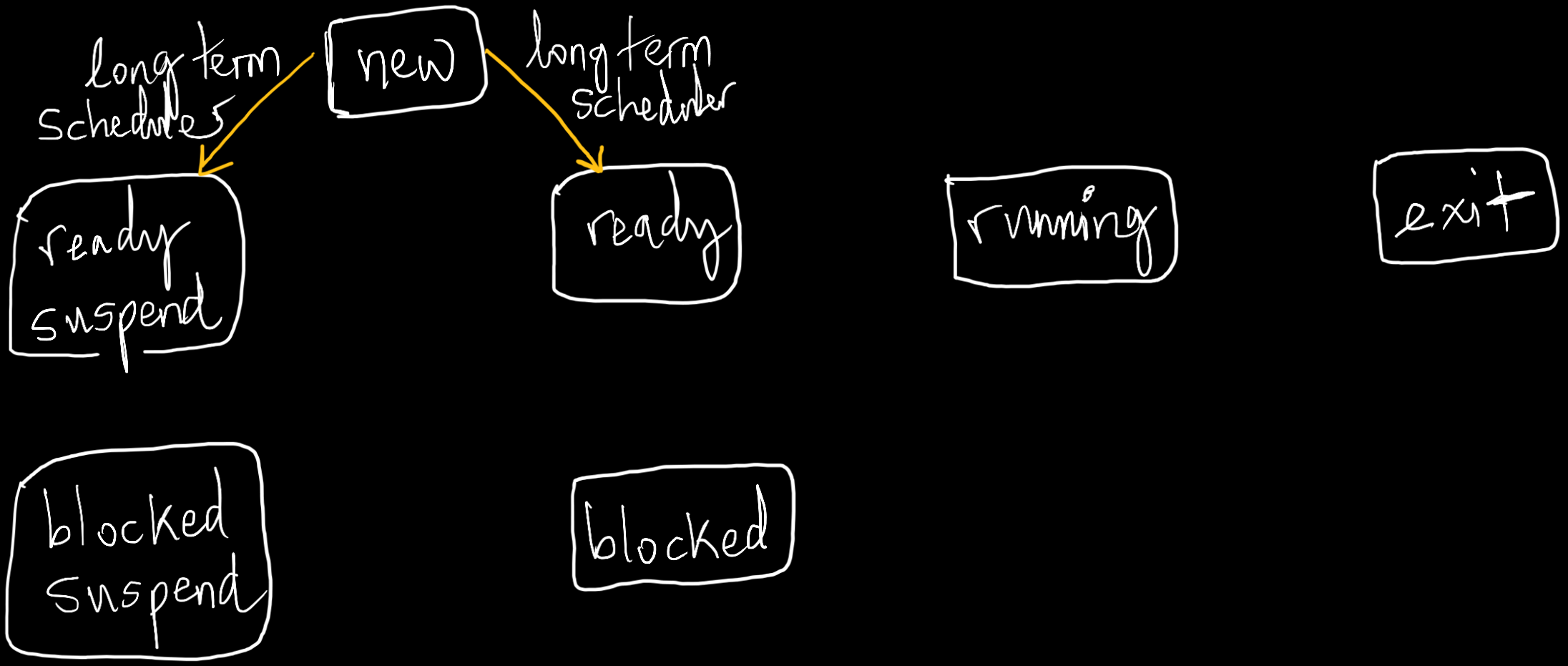
If **foo** is a command, the **man** command will display **foo**'s manual in a Unix shell by executing **man foo** from the shell's command prompt. So, for example, executing **man man** in a shell runs a program -- i.e. creates a process -- that displays the manual for **man**. The **echo** command will print the string it is given -- in this case, the process id (stored in the variable $) for the foreground shell process. And, **ls** lists the files and directories in the present working directory, while "**;**" is simply a delimiter to separate commands at the command prompt.

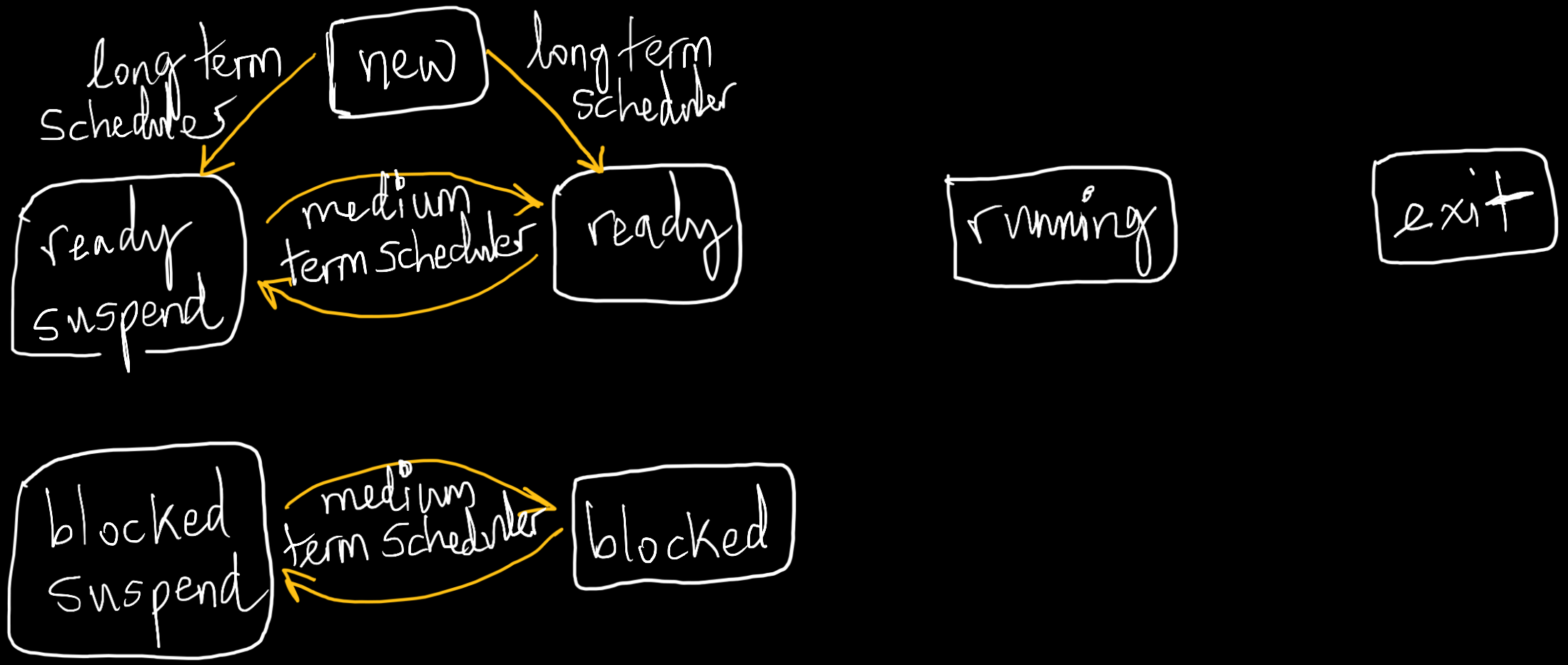
Therefore, issuing **man man** at the prompt creates a process, then **man echo** creates another process, **echo** creates a 3rd process and **ls -al** creates a fourth process.

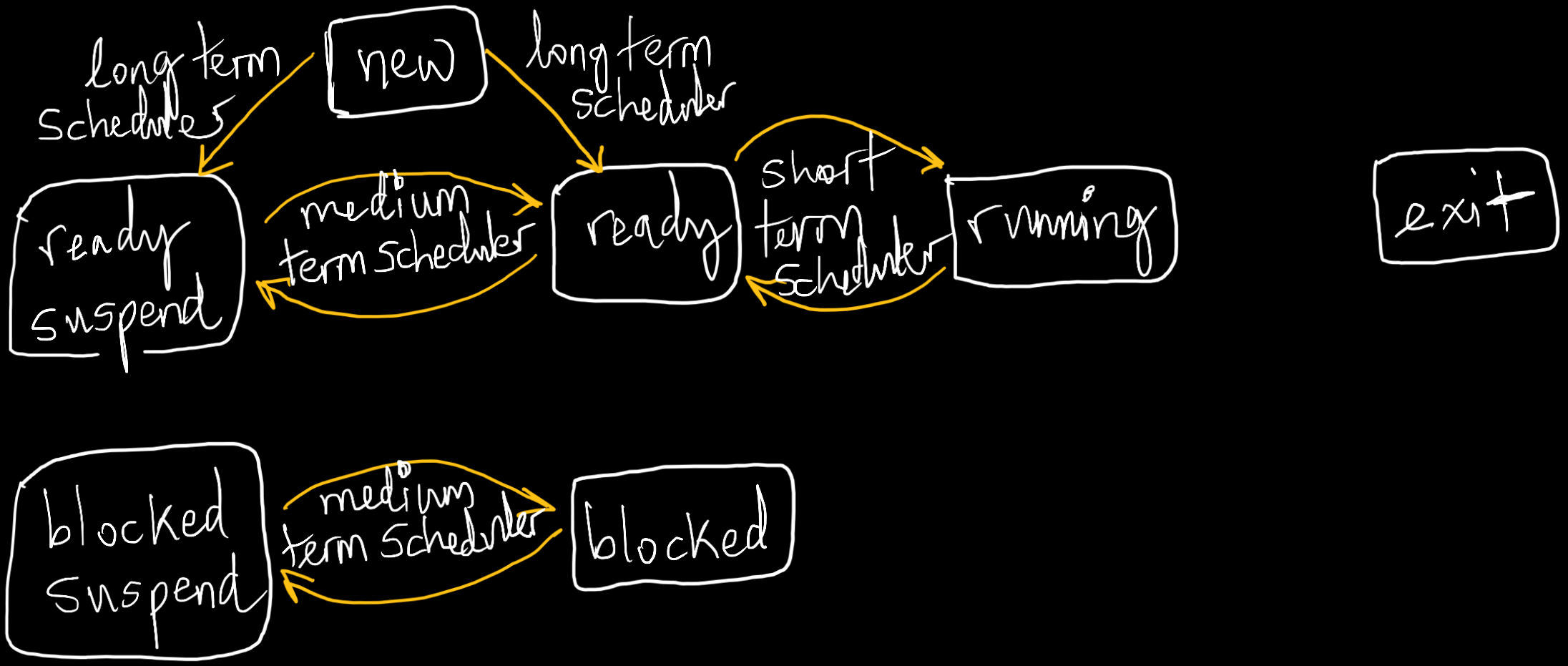
See the "Introduction to Linux shells and commands" instructions from tutorial 1 for **man** and **ls.** See "Working with Linux processes" in tutorial 2 for **echo** and delimiting commands at the command prompt using **;**.

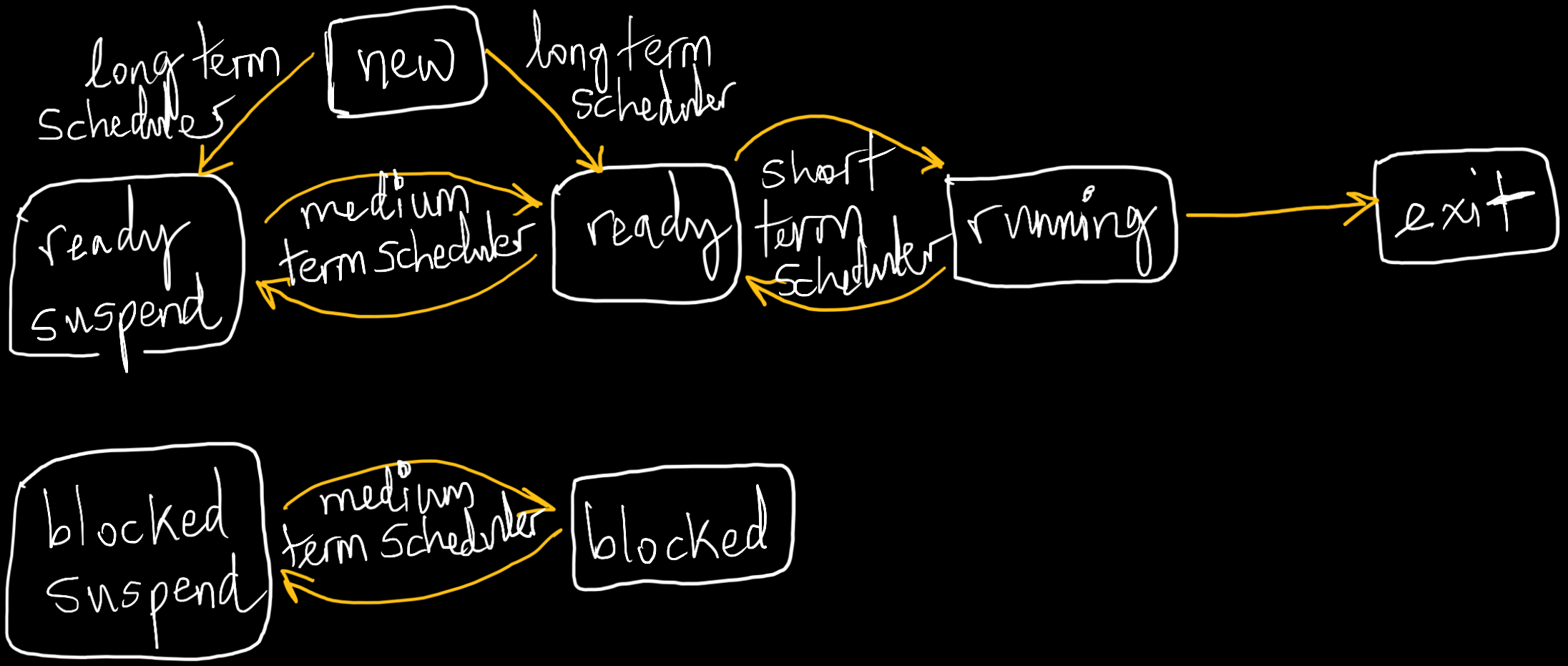

long term Scheduler
new
long term Scheduler
ready suspend
medium term Scheduler
ready
short term Scheduler
running
exit
blocked suspend
medium term Scheduler
blocked


new
ready suspend
ready
running
exit
blocked suspend
blocked


long term Scheduler
new
long term Scheduler
ready suspend
ready
running
exit
blocked suspend
blocked


long term Scheduler
new
long term Scheduler
ready suspend
medium term Scheduler
ready
running
exit
blocked suspend
medium term Scheduler
blocked


long term Scheduler
new
long term Scheduler
ready suspend
medium term Scheduler
ready
short term Scheduler
running
exit
blocked suspend
medium term Scheduler
blocked


long term Scheduler
new
long term Scheduler
ready suspend
medium term Scheduler
ready
short term Scheduler
running
exit
blocked suspend
medium term Scheduler
blocked