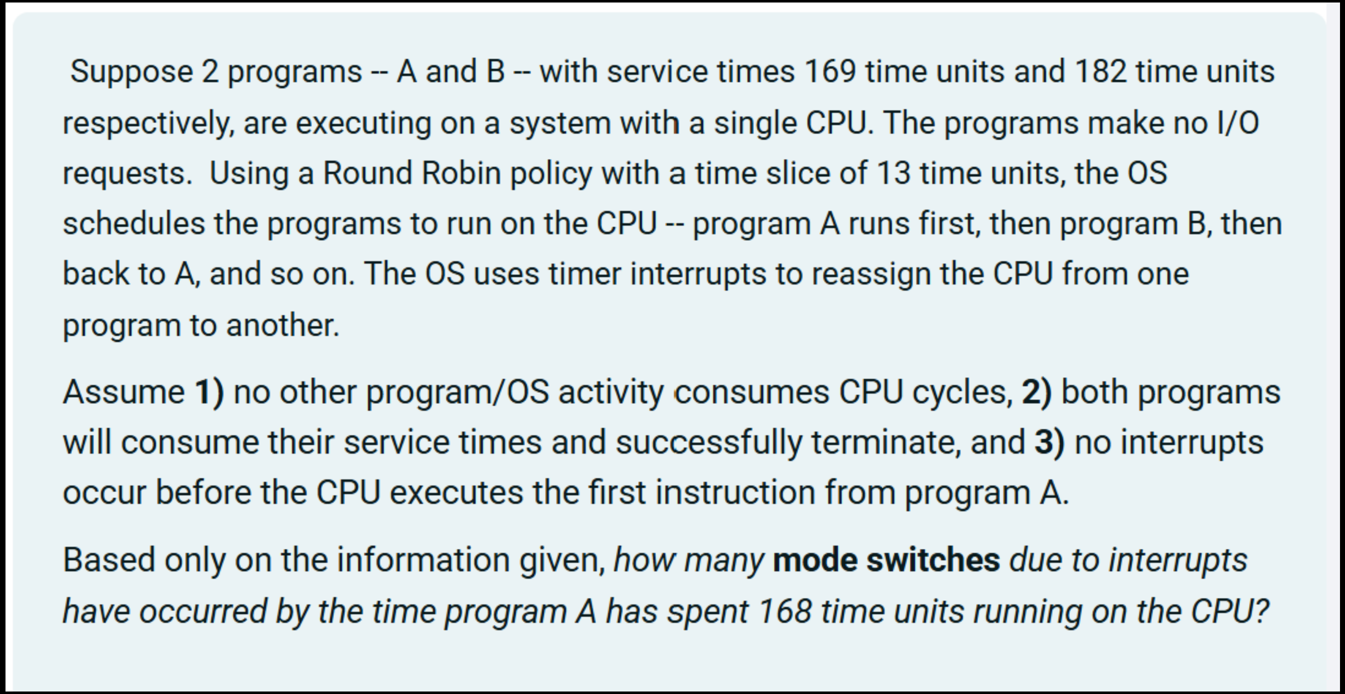
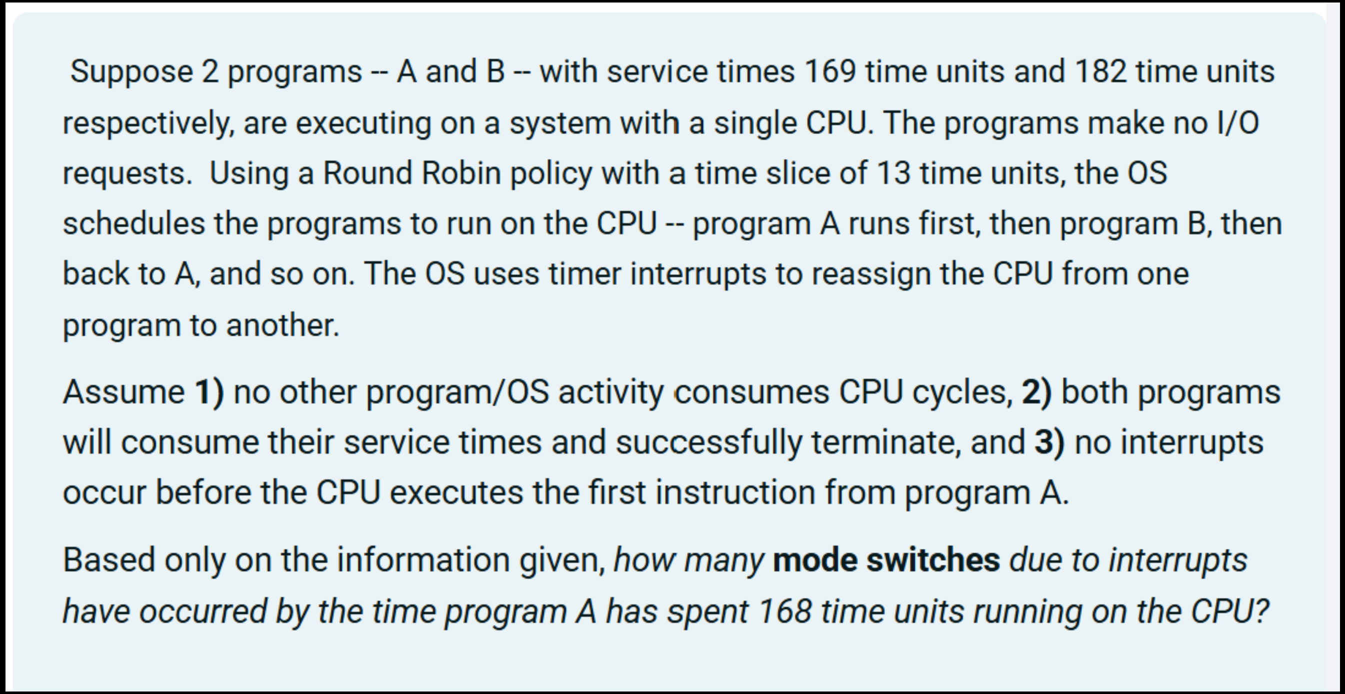


Suppose 2 programs -- A and B -- with service times 169 time units and 182 time units respectively, are executing on a system with a single CPU. The programs make no I/O requests. Using a Round Robin policy with a time slice of 13 time units, the OS schedules the programs to run on the CPU -- program A runs first, then program B, then back to A, and so on. The OS uses timer interrupts to reassign the CPU from one program to another.

Assume **1)** no other program/OS activity consumes CPU cycles, **2)** both programs will consume their service times and successfully terminate, and **3)** no interrupts occur before the CPU executes the first instruction from program A.

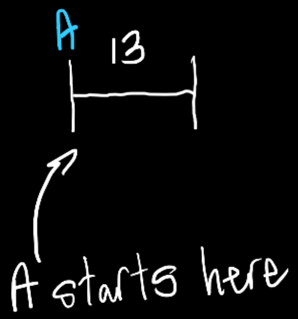
Based only on the information given, *how many* **mode switches** *due to interrupts have occurred by the time program A has spent 168 time units running on the CPU?*

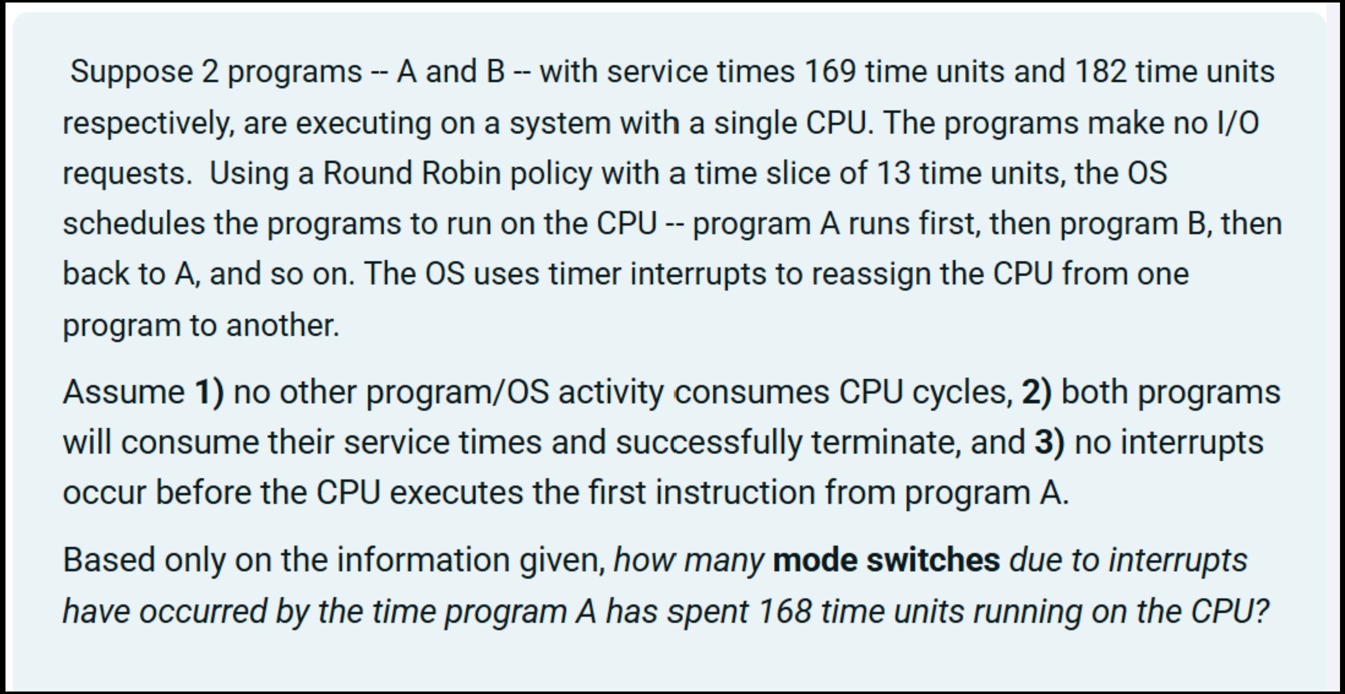


Suppose 2 programs -- A and B -- with service times 169 time units and 182 time units respectively, are executing on a system with a single CPU. The programs make no I/O requests. Using a Round Robin policy with a time slice of 13 time units, the OS schedules the programs to run on the CPU -- program A runs first, then program B, then back to A, and so on. The OS uses timer interrupts to reassign the CPU from one program to another.

Assume **1)** no other program/OS activity consumes CPU cycles, **2)** both programs will consume their service times and successfully terminate, and **3)** no interrupts occur before the CPU executes the first instruction from program A.

Based only on the information given, *how many* **mode switches** *due to interrupts have occurred by the time program A has spent 168 time units running on the CPU?*

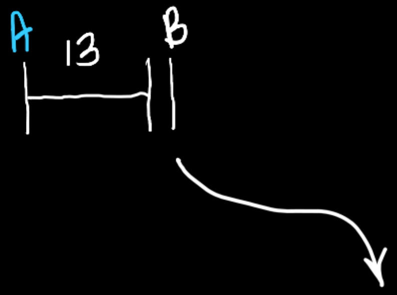




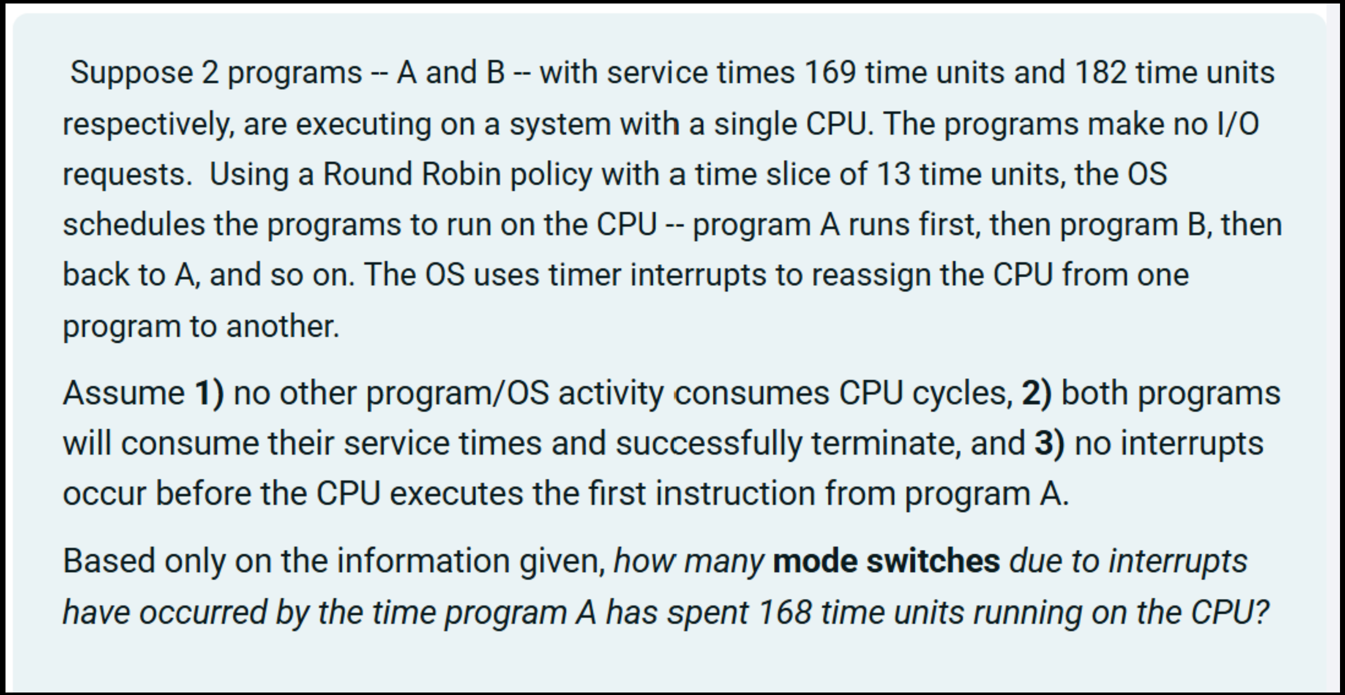
Suppose 2 programs -- A and B -- with service times 169 time units and 182 time units respectively, are executing on a system with a single CPU. The programs make no I/O requests. Using a Round Robin policy with a time slice of 13 time units, the OS schedules the programs to run on the CPU -- program A runs first, then program B, then back to A, and so on. The OS uses timer interrupts to reassign the CPU from one program to another.

Assume **1)** no other program/OS activity consumes CPU cycles, **2)** both programs will consume their service times and successfully terminate, and **3)** no interrupts occur before the CPU executes the first instruction from program A.

Based only on the information given, *how many* **mode switches** *due to interrupts have occurred by the time program A has spent 168 time units running on the CPU?*



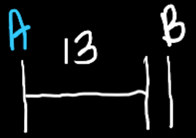
Timer interrupt
for long-term scheduler
to switch A with B on
the CPU.



Suppose 2 programs -- A and B -- with service times 169 time units and 182 time units respectively, are executing on a system with a single CPU. The programs make no I/O requests. Using a Round Robin policy with a time slice of 13 time units, the OS schedules the programs to run on the CPU -- program A runs first, then program B, then back to A, and so on. The OS uses timer interrupts to reassign the CPU from one program to another.

Assume **1)** no other program/OS activity consumes CPU cycles, **2)** both programs will consume their service times and successfully terminate, and **3)** no interrupts occur before the CPU executes the first instruction from program A.

Based only on the information given, *how many* **mode switches** *due to interrupts have occurred by the time program A has spent 168 time units running on the CPU?*

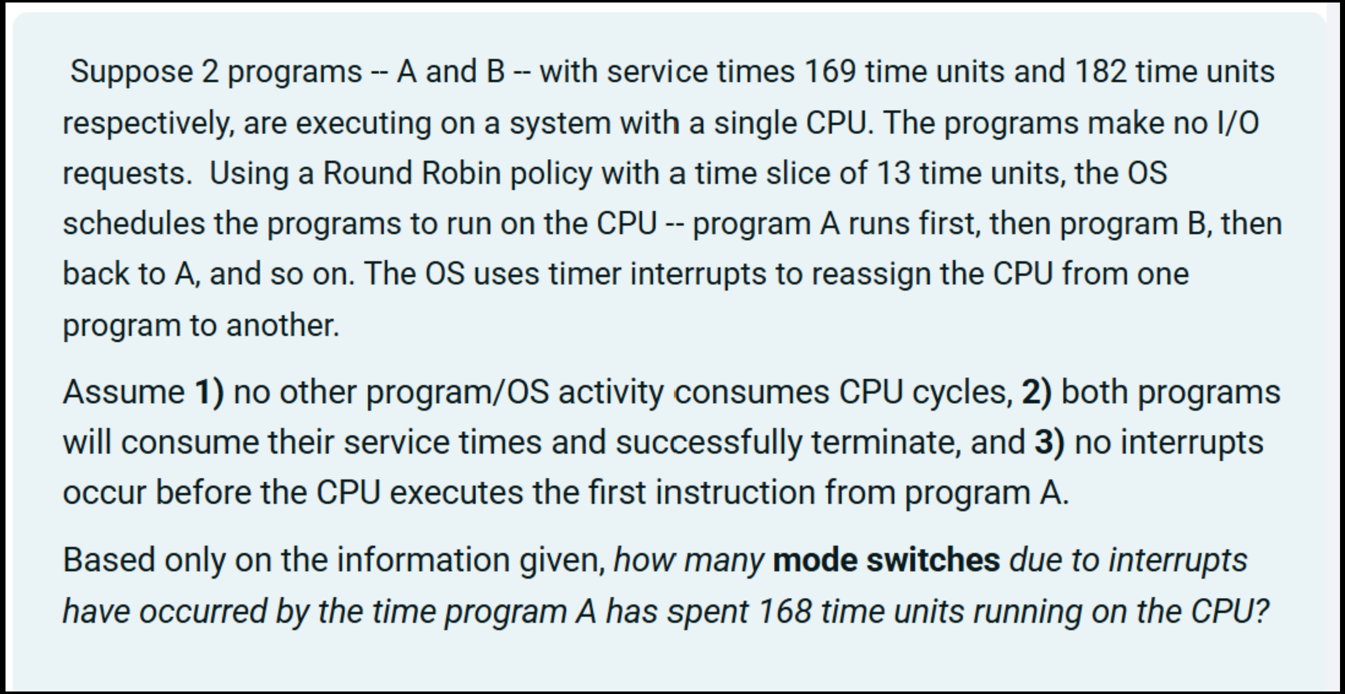


Timer interrupt for long-term scheduler to switch A with B on the CPU. →

2 mode switches:

User → Kernel mode

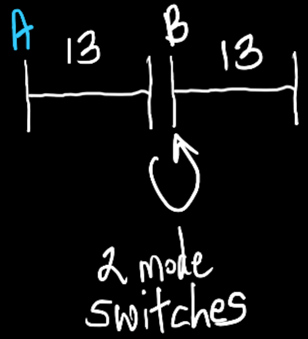
Kernel → User mode

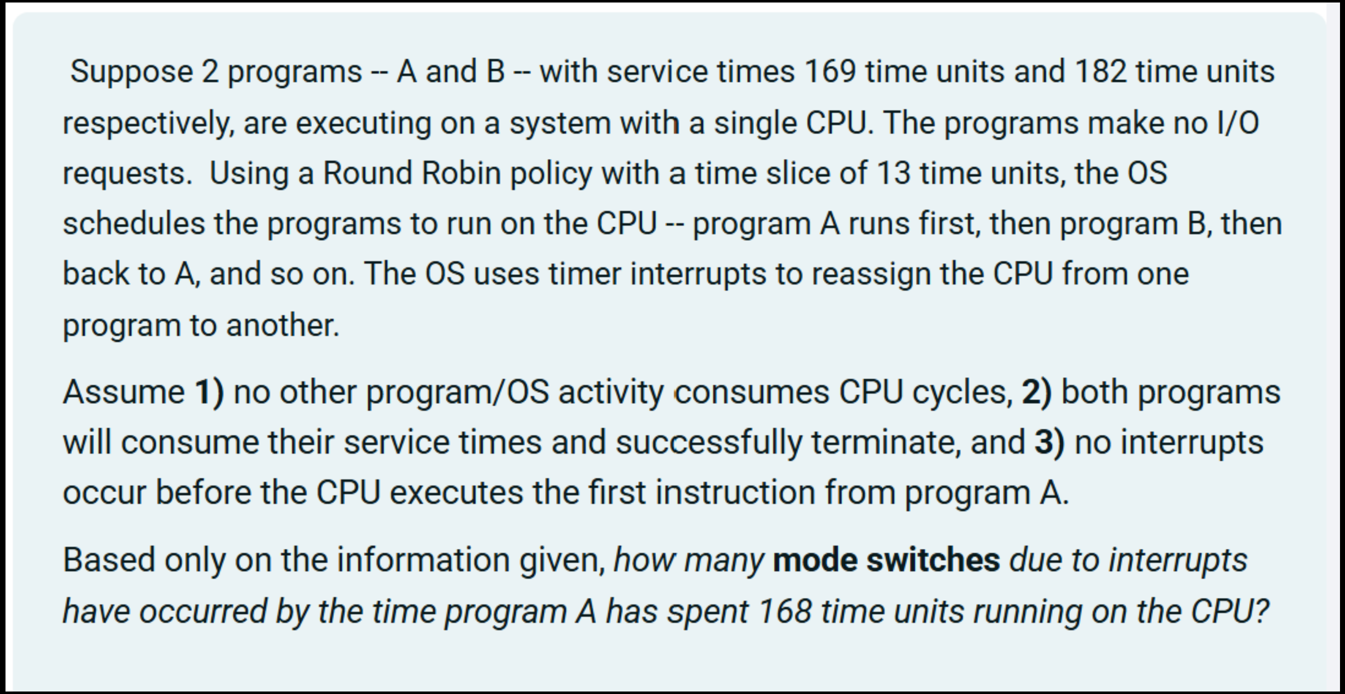


Suppose 2 programs -- A and B -- with service times 169 time units and 182 time units respectively, are executing on a system with a single CPU. The programs make no I/O requests. Using a Round Robin policy with a time slice of 13 time units, the OS schedules the programs to run on the CPU -- program A runs first, then program B, then back to A, and so on. The OS uses timer interrupts to reassign the CPU from one program to another.

Assume **1)** no other program/OS activity consumes CPU cycles, **2)** both programs will consume their service times and successfully terminate, and **3)** no interrupts occur before the CPU executes the first instruction from program A.

Based only on the information given, *how many* **mode switches** *due to interrupts have occurred by the time program A has spent 168 time units running on the CPU?*

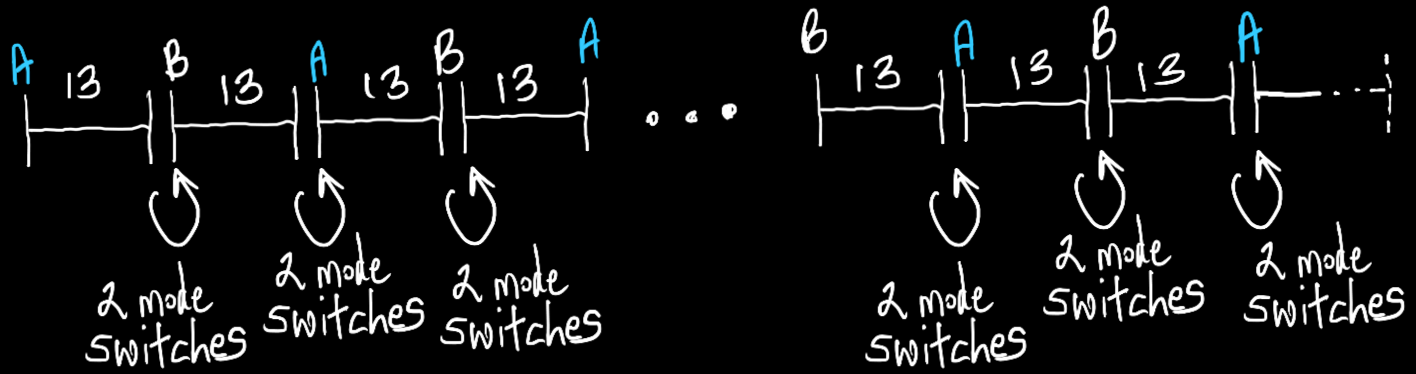


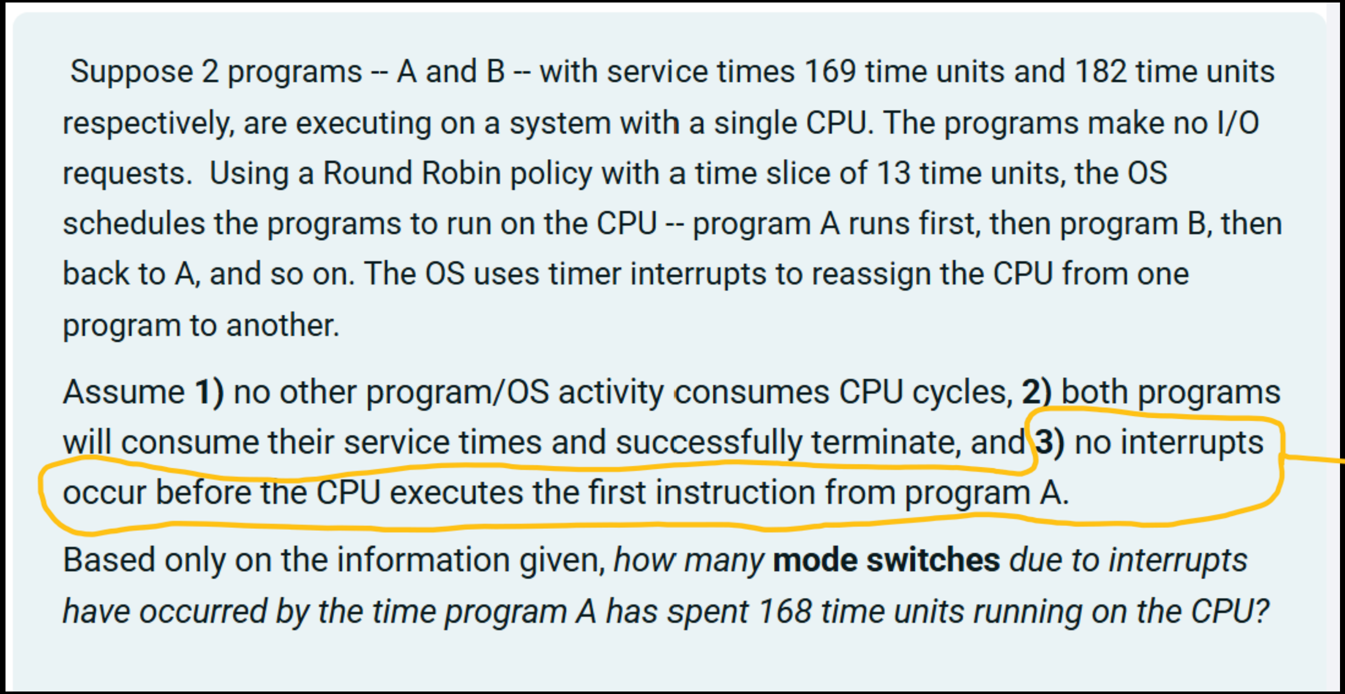


Suppose 2 programs -- A and B -- with service times 169 time units and 182 time units respectively, are executing on a system with a single CPU. The programs make no I/O requests. Using a Round Robin policy with a time slice of 13 time units, the OS schedules the programs to run on the CPU -- program A runs first, then program B, then back to A, and so on. The OS uses timer interrupts to reassign the CPU from one program to another.

Assume **1)** no other program/OS activity consumes CPU cycles, **2)** both programs will consume their service times and successfully terminate, and **3)** no interrupts occur before the CPU executes the first instruction from program A.

Based only on the information given, *how many* **mode switches** *due to interrupts have occurred by the time program A has spent 168 time units running on the CPU?*

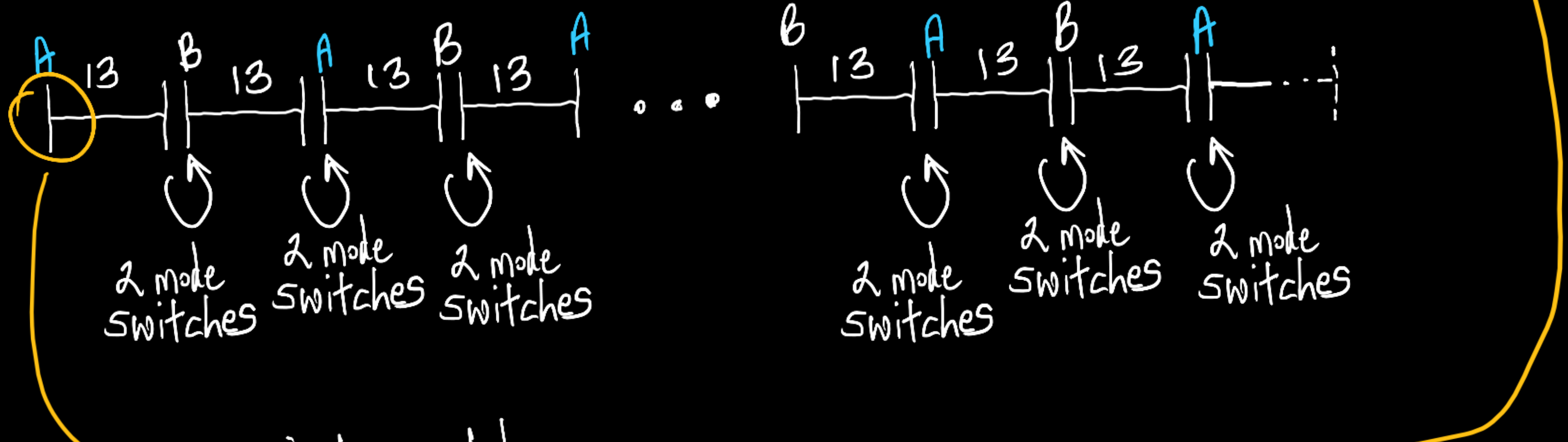




Suppose 2 programs -- A and B -- with service times 169 time units and 182 time units respectively, are executing on a system with a single CPU. The programs make no I/O requests. Using a Round Robin policy with a time slice of 13 time units, the OS schedules the programs to run on the CPU -- program A runs first, then program B, then back to A, and so on. The OS uses timer interrupts to reassign the CPU from one program to another.

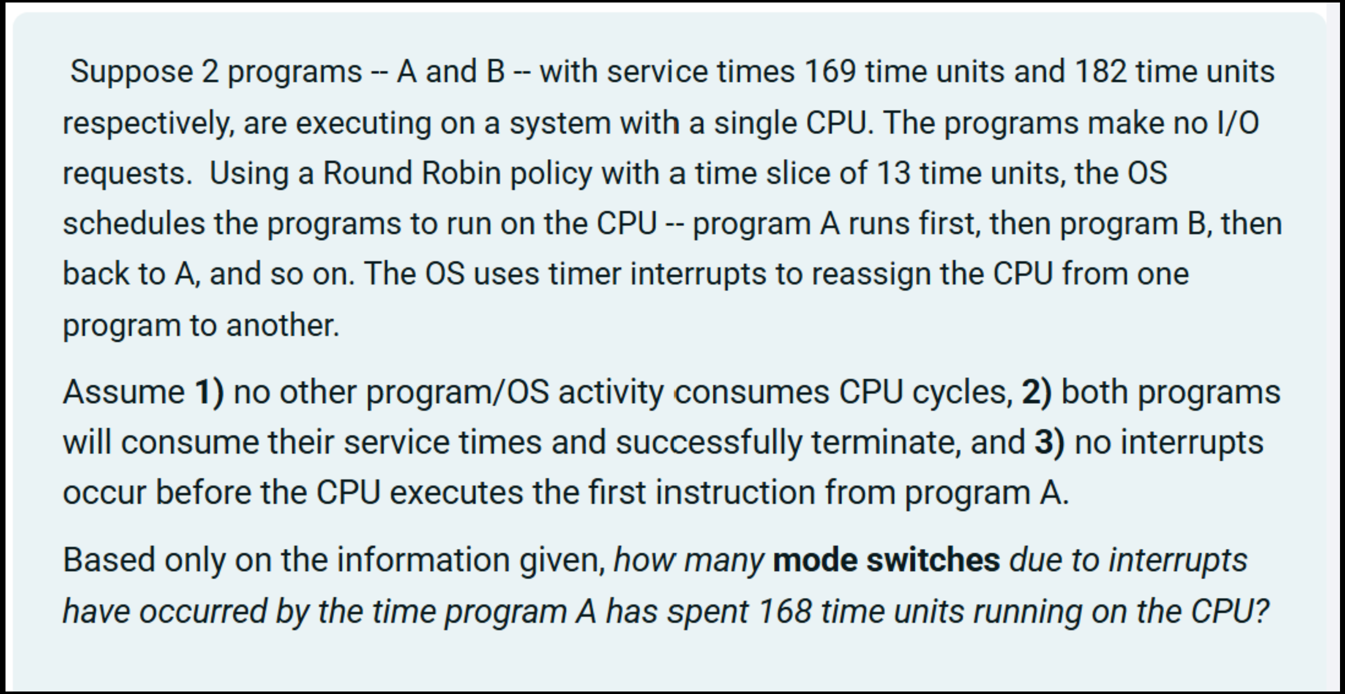
Assume **1)** no other program/OS activity consumes CPU cycles, **2)** both programs will consume their service times and successfully terminate, and **3)** no interrupts occur before the CPU executes the first instruction from program A.

Based only on the information given, *how many* **mode switches** *due to interrupts have occurred by the time program A has spent 168 time units running on the CPU?*



No interrupt!

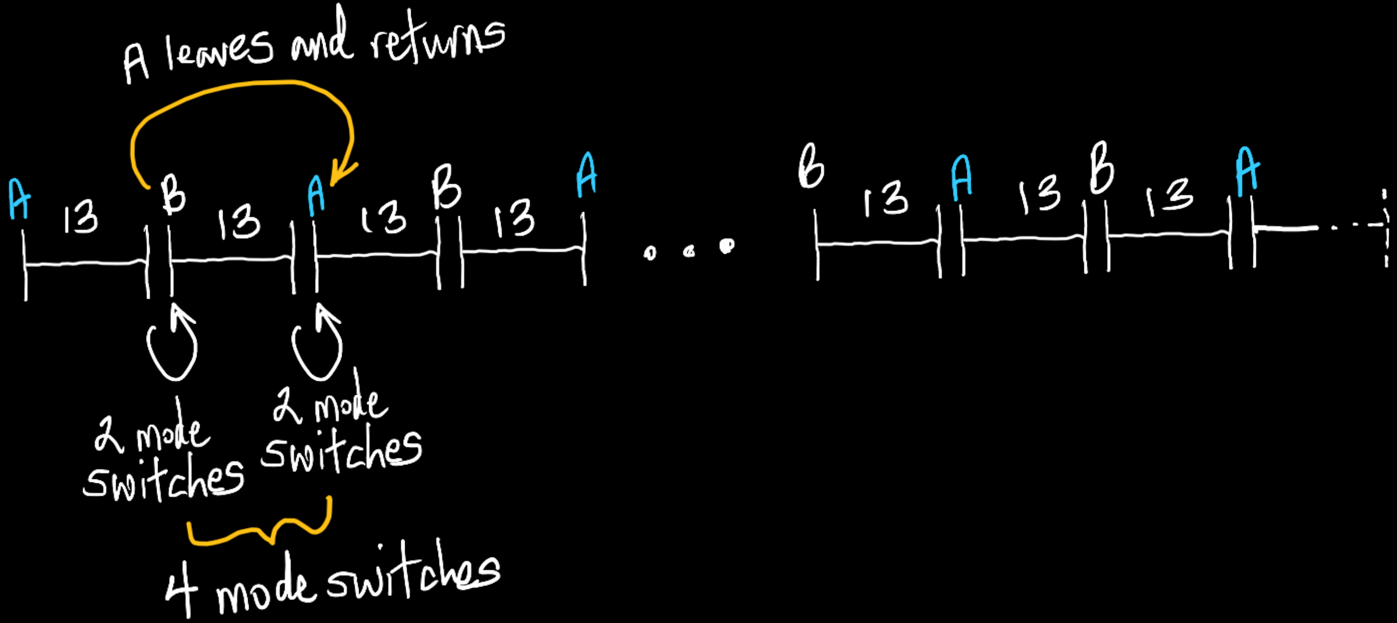
No mode switches!

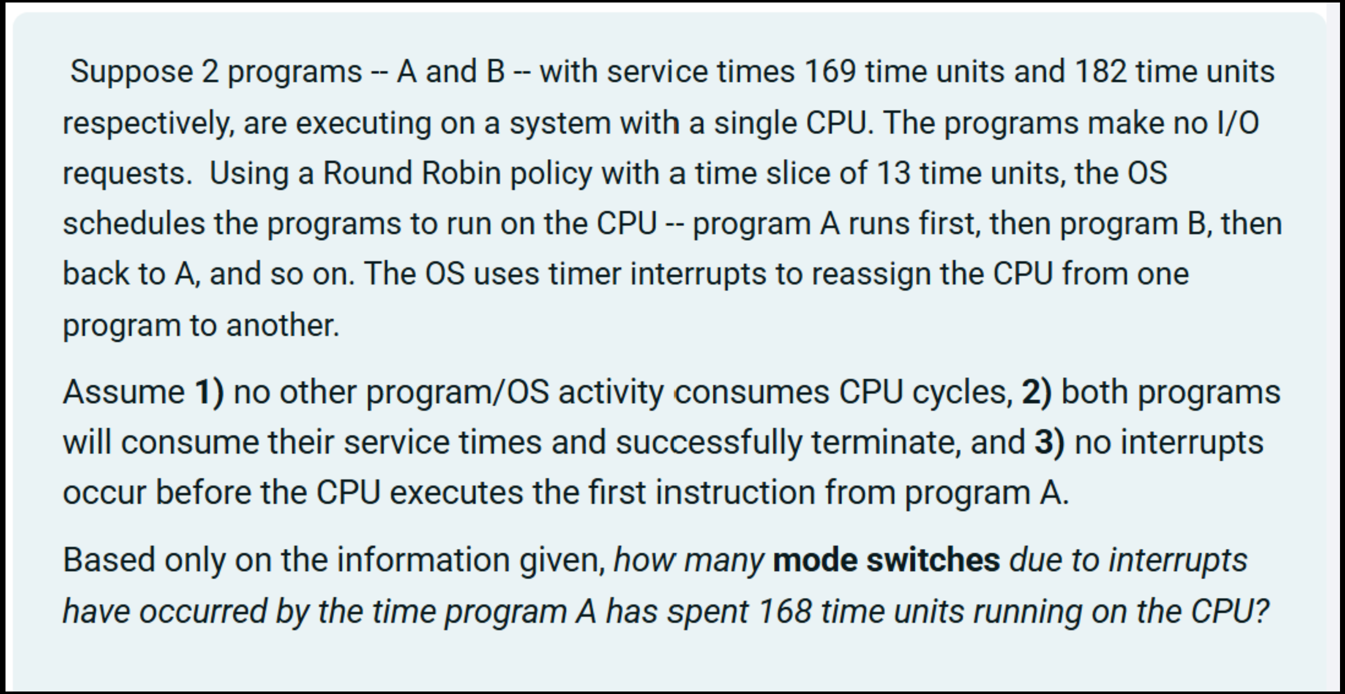


Suppose 2 programs -- A and B -- with service times 169 time units and 182 time units respectively, are executing on a system with a single CPU. The programs make no I/O requests. Using a Round Robin policy with a time slice of 13 time units, the OS schedules the programs to run on the CPU -- program A runs first, then program B, then back to A, and so on. The OS uses timer interrupts to reassign the CPU from one program to another.

Assume **1)** no other program/OS activity consumes CPU cycles, **2)** both programs will consume their service times and successfully terminate, and **3)** no interrupts occur before the CPU executes the first instruction from program A.

Based only on the information given, *how many* **mode switches** *due to interrupts have occurred by the time program A has spent 168 time units running on the CPU?*

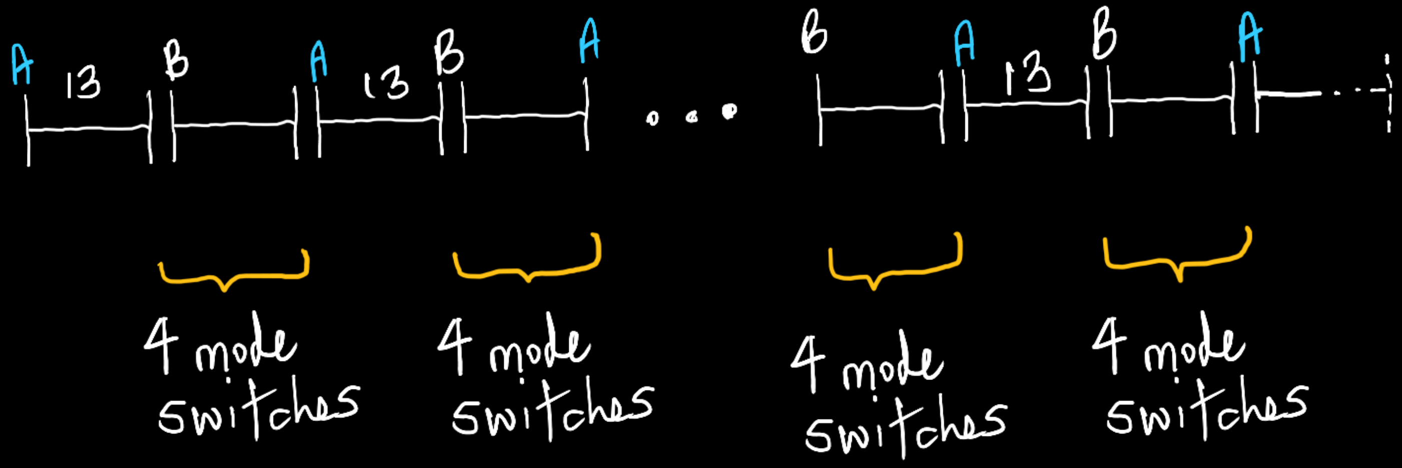


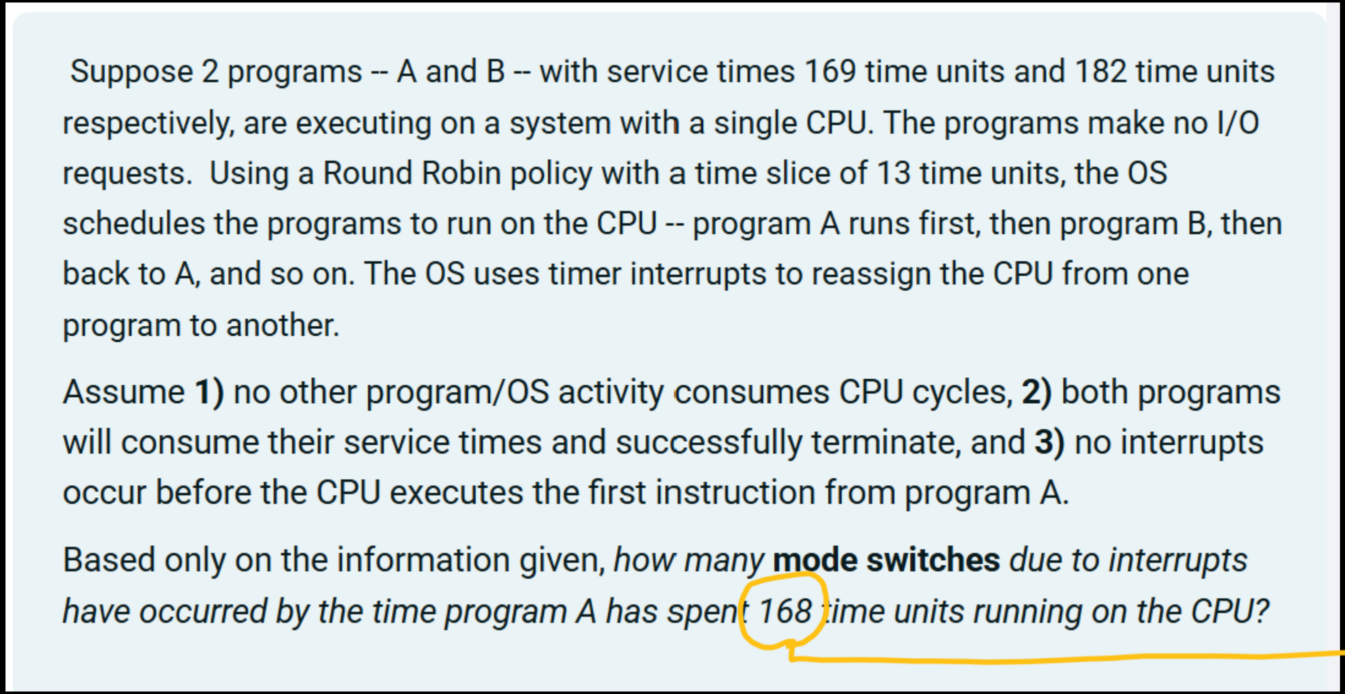


Suppose 2 programs -- A and B -- with service times 169 time units and 182 time units respectively, are executing on a system with a single CPU. The programs make no I/O requests. Using a Round Robin policy with a time slice of 13 time units, the OS schedules the programs to run on the CPU -- program A runs first, then program B, then back to A, and so on. The OS uses timer interrupts to reassign the CPU from one program to another.

Assume **1)** no other program/OS activity consumes CPU cycles, **2)** both programs will consume their service times and successfully terminate, and **3)** no interrupts occur before the CPU executes the first instruction from program A.

Based only on the information given, *how many* **mode switches** *due to interrupts have occurred by the time program A has spent 168 time units running on the CPU?*

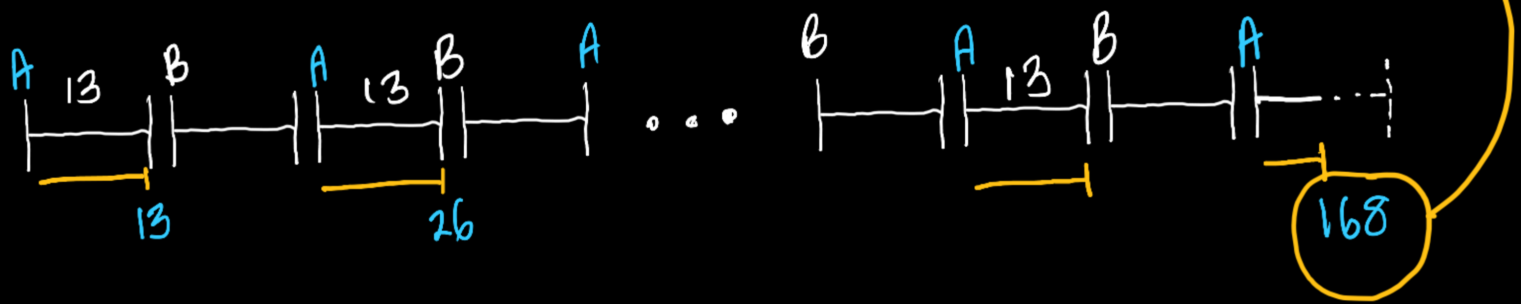


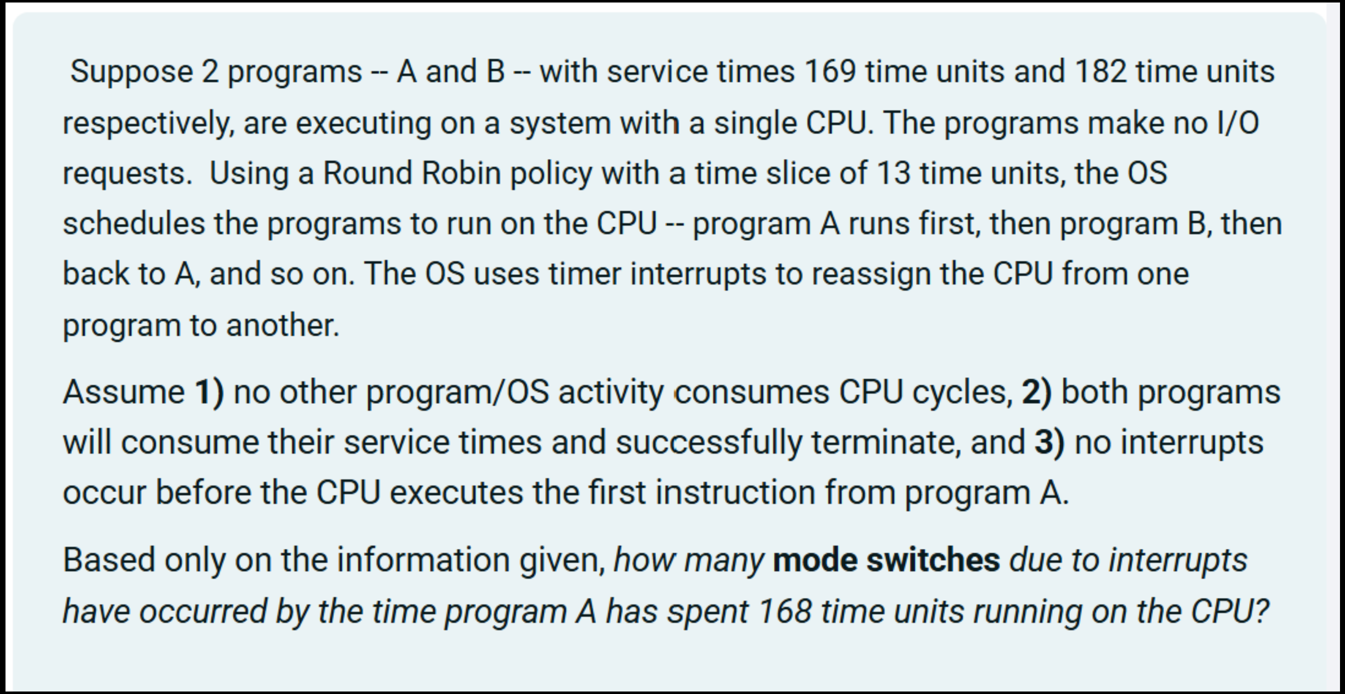


Suppose 2 programs -- A and B -- with service times 169 time units and 182 time units respectively, are executing on a system with a single CPU. The programs make no I/O requests. Using a Round Robin policy with a time slice of 13 time units, the OS schedules the programs to run on the CPU -- program A runs first, then program B, then back to A, and so on. The OS uses timer interrupts to reassign the CPU from one program to another.

Assume **1)** no other program/OS activity consumes CPU cycles, **2)** both programs will consume their service times and successfully terminate, and **3)** no interrupts occur before the CPU executes the first instruction from program A.

Based only on the information given, *how many* **mode switches** *due to interrupts have occurred by the time program A has spent 168 time units running on the CPU?*

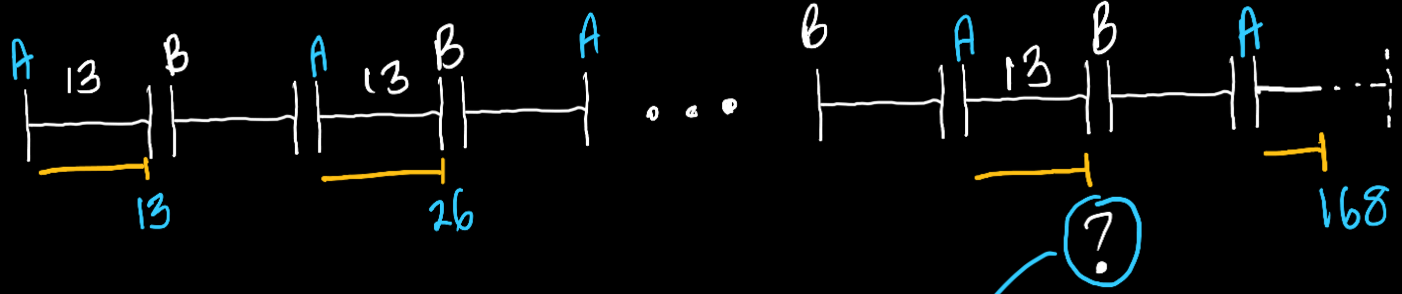




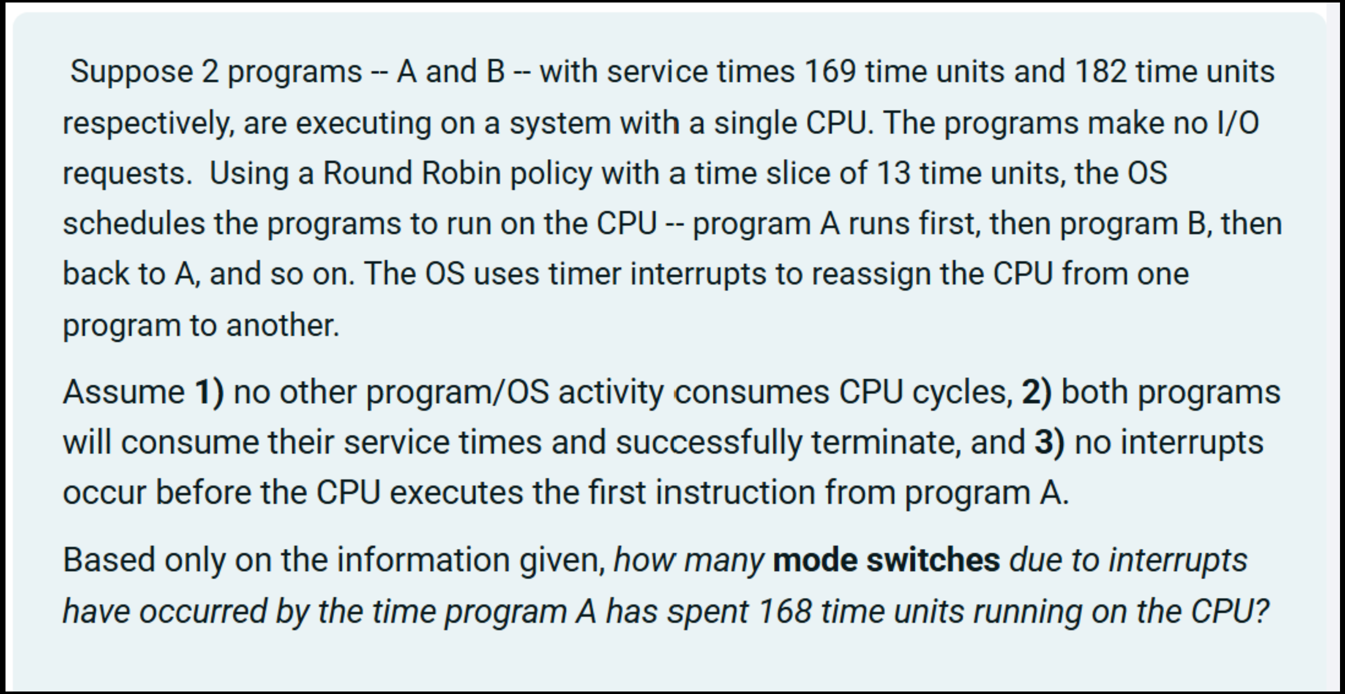
Suppose 2 programs -- A and B -- with service times 169 time units and 182 time units respectively, are executing on a system with a single CPU. The programs make no I/O requests. Using a Round Robin policy with a time slice of 13 time units, the OS schedules the programs to run on the CPU -- program A runs first, then program B, then back to A, and so on. The OS uses timer interrupts to reassign the CPU from one program to another.

Assume **1)** no other program/OS activity consumes CPU cycles, **2)** both programs will consume their service times and successfully terminate, and **3)** no interrupts occur before the CPU executes the first instruction from program A.

Based only on the information given, *how many* **mode switches** *due to interrupts have occurred by the time program A has spent 168 time units running on the CPU?*



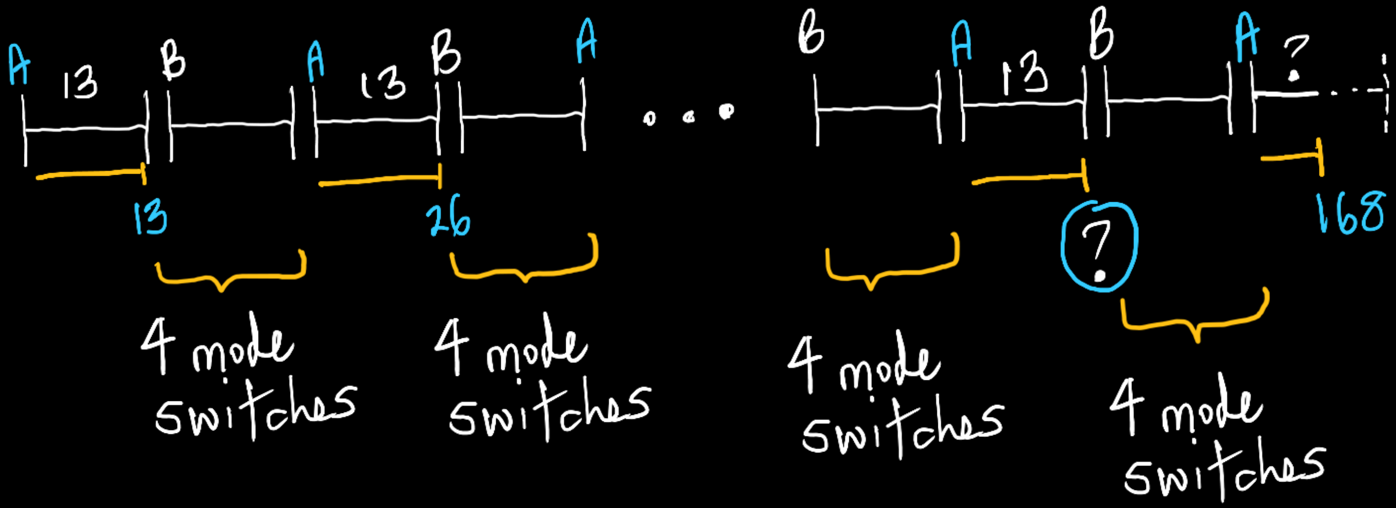
What is the largest number of "slices" of length 13 we can fit inside 168?

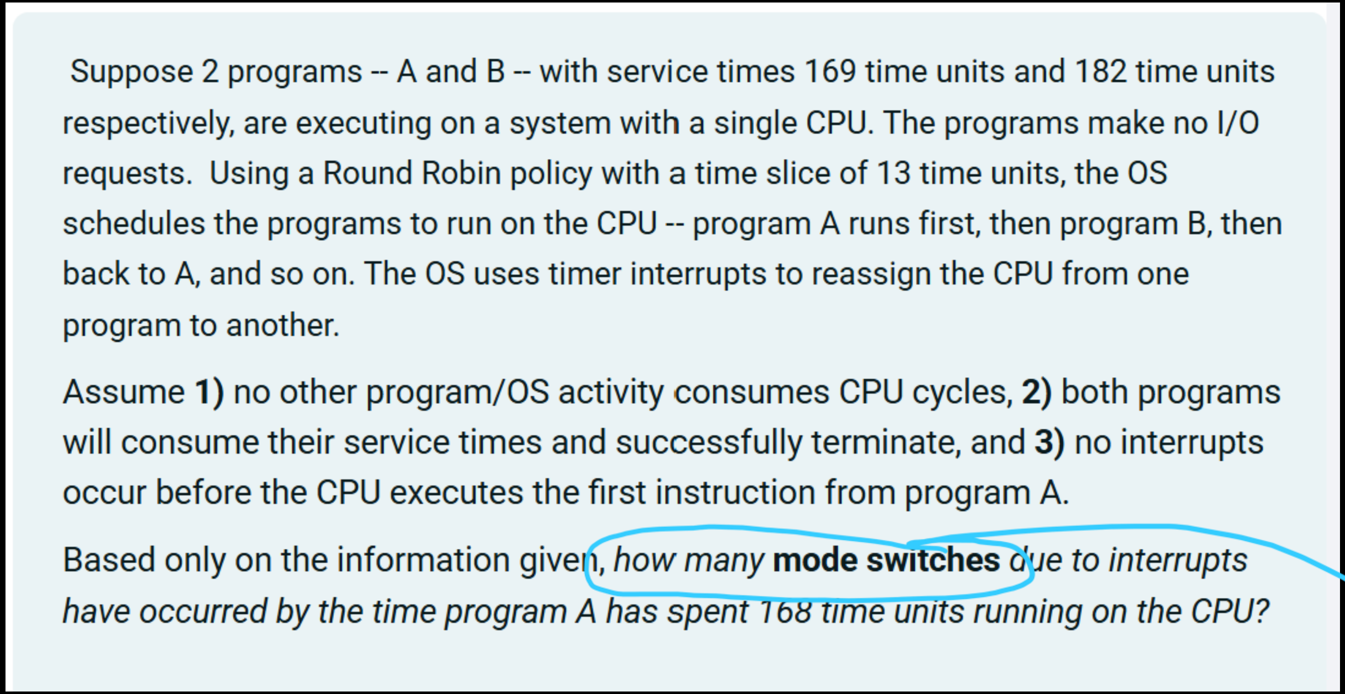


Suppose 2 programs -- A and B -- with service times 169 time units and 182 time units respectively, are executing on a system with a single CPU. The programs make no I/O requests. Using a Round Robin policy with a time slice of 13 time units, the OS schedules the programs to run on the CPU -- program A runs first, then program B, then back to A, and so on. The OS uses timer interrupts to reassign the CPU from one program to another.

Assume **1)** no other program/OS activity consumes CPU cycles, **2)** both programs will consume their service times and successfully terminate, and **3)** no interrupts occur before the CPU executes the first instruction from program A.

Based only on the information given, *how many* **mode switches** *due to interrupts have occurred by the time program A has spent 168 time units running on the CPU?*





Suppose 2 programs -- A and B -- with service times 169 time units and 182 time units respectively, are executing on a system with a single CPU. The programs make no I/O requests. Using a Round Robin policy with a time slice of 13 time units, the OS schedules the programs to run on the CPU -- program A runs first, then program B, then back to A, and so on. The OS uses timer interrupts to reassign the CPU from one program to another.

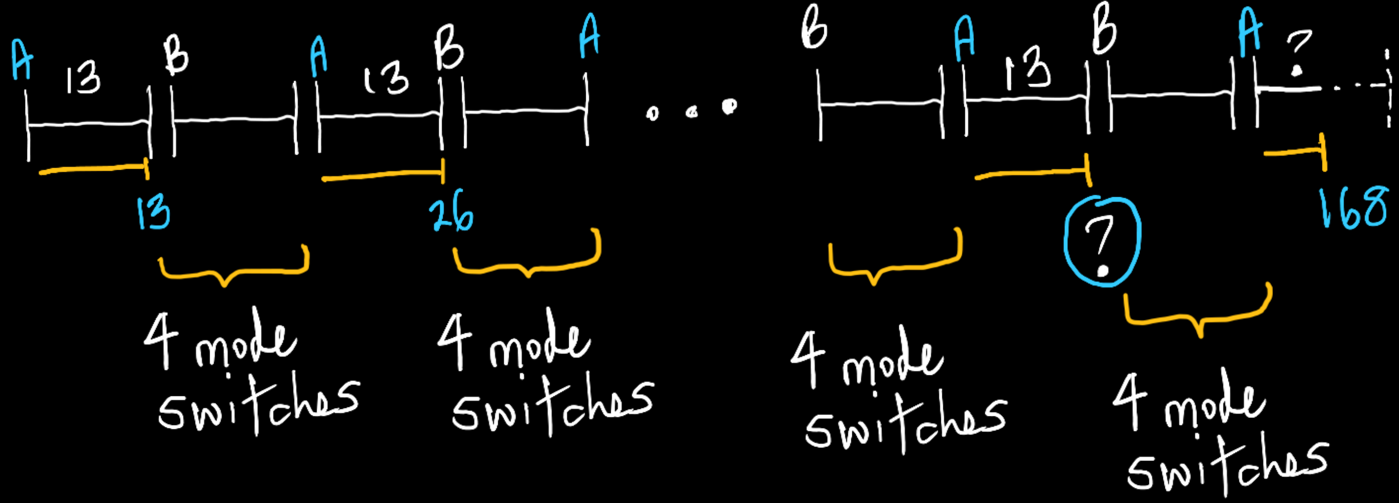
Assume **1)** no other program/OS activity consumes CPU cycles, **2)** both programs will consume their service times and successfully terminate, and **3)** no interrupts occur before the CPU executes the first instruction from program A.

Based only on the information given, *how many* ***mode switches*** *due to interrupts have occurred by the time program A has spent 168 time units running on the CPU?*

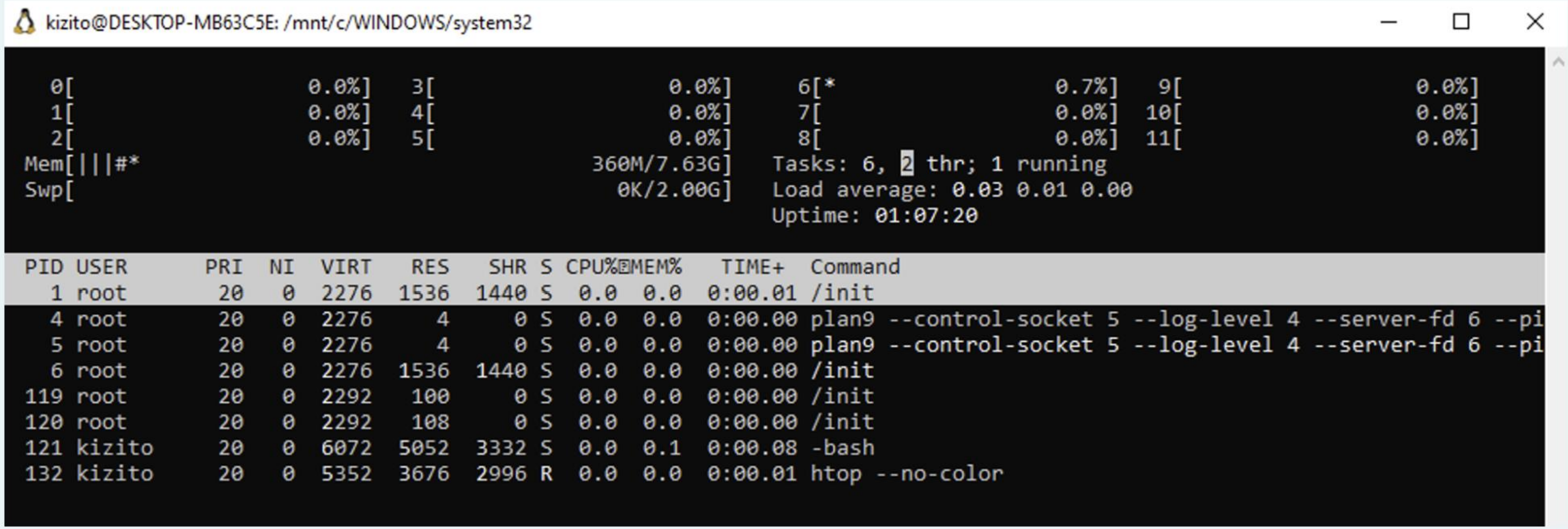
Number of mode switches

= (Number of "slices" of length 13 we can place in 168) × 4

$$= \left\lfloor \frac{168}{13} \right\rfloor \times 4 = 12 \times 4 = 48.$$



Using the information in the table below, what is the parent process ID for the process that printed this table to the Linux shell?

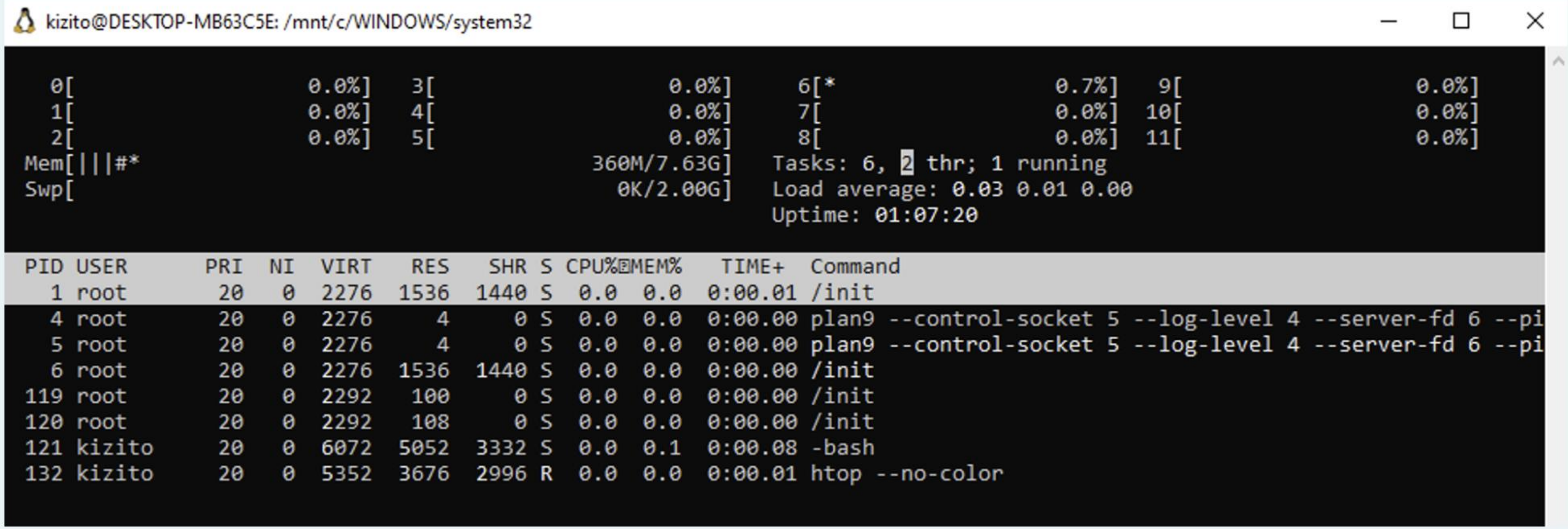


kizito@DESKTOP-MB63C5E: /mnt/c/WINDOWS/system32

```
  0[                  0.0%]   3[                  0.0%]    6[*                 0.7%]   9[                  0.0%]
  1[                  0.0%]   4[                  0.0%]    7[                  0.0%]  10[                  0.0%]
  2[                  0.0%]   5[                  0.0%]    8[                  0.0%]  11[                  0.0%]
Mem[|||#*                               360M/7.63G]   Tasks: 6, 2 thr; 1 running
Swp[                                      0K/2.00G]   Load average: 0.03 0.01 0.00
                                                      Uptime: 01:07:20

PID USER       PRI  NI  VIRT   RES   SHR S CPU%▯MEM%   TIME+  Command
  1 root        20   0  2276  1536  1440 S  0.0  0.0  0:00.01 /init
  4 root        20   0  2276     4     0 S  0.0  0.0  0:00.00 plan9 --control-socket 5 --log-level 4 --server-fd 6 --pi
  5 root        20   0  2276     4     0 S  0.0  0.0  0:00.00 plan9 --control-socket 5 --log-level 4 --server-fd 6 --pi
  6 root        20   0  2276  1536  1440 S  0.0  0.0  0:00.00 /init
119 root        20   0  2292   100     0 S  0.0  0.0  0:00.00 /init
120 root        20   0  2292   108     0 S  0.0  0.0  0:00.00 /init
121 kizito      20   0  6072  5052  3332 S  0.0  0.1  0:00.08 -bash
132 kizito      20   0  5352  3676  2996 R  0.0  0.0  0:00.01 htop --no-color
```

Answer:

Using the information in the table below, what is the parent process ID for the process that printed this table to the Linux shell?

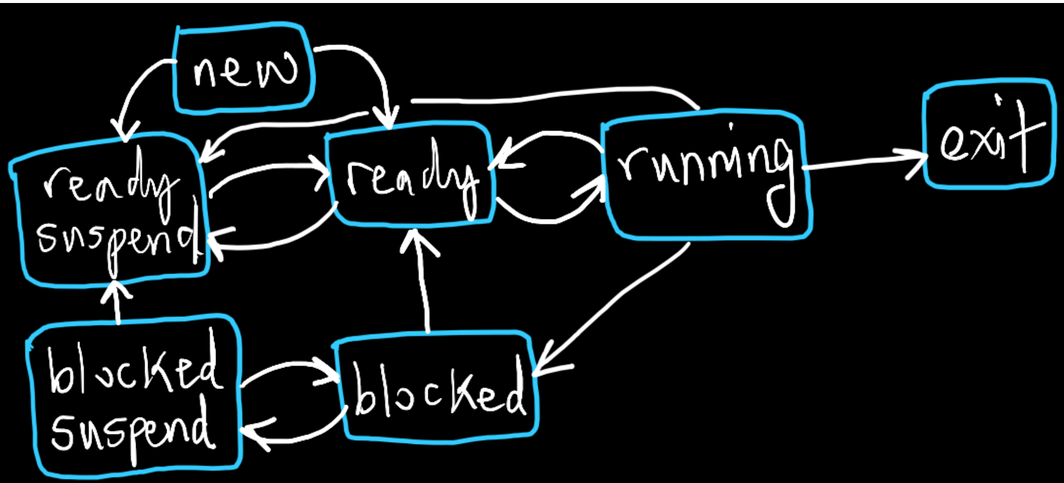


kizito@DESKTOP-MB63C5E: /mnt/c/WINDOWS/system32

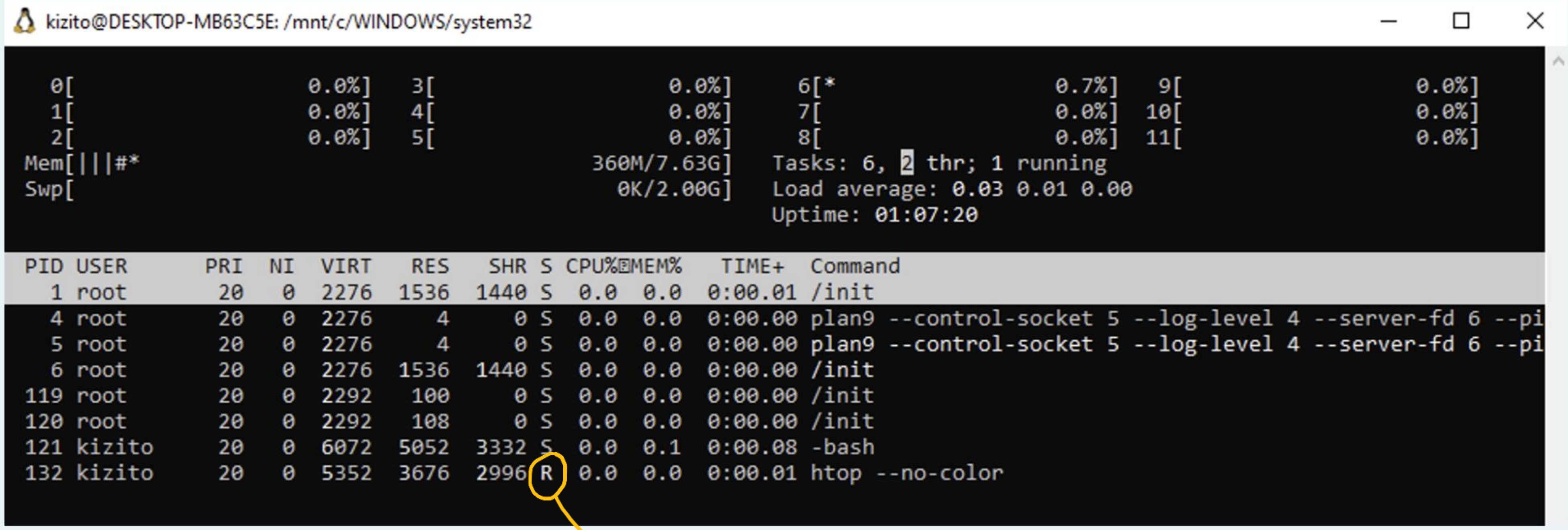
```
  0[                   0.0%]   3[                   0.0%]    6[*                  0.7%]   9[                   0.0%]
  1[                   0.0%]   4[                   0.0%]    7[                   0.0%]  10[                   0.0%]
  2[                   0.0%]   5[                   0.0%]    8[                   0.0%]  11[                   0.0%]
Mem[|||#*                                    360M/7.63G]  Tasks: 6, 2 thr; 1 running
Swp[                                           0K/2.00G]  Load average: 0.03 0.01 0.00
                                                          Uptime: 01:07:20

PID USER       PRI  NI  VIRT   RES   SHR S CPU%▯MEM%   TIME+  Command
  1 root        20   0  2276  1536  1440 S  0.0  0.0  0:00.01 /init
  4 root        20   0  2276     4     0 S  0.0  0.0  0:00.00 plan9 --control-socket 5 --log-level 4 --server-fd 6 --pi
  5 root        20   0  2276     4     0 S  0.0  0.0  0:00.00 plan9 --control-socket 5 --log-level 4 --server-fd 6 --pi
  6 root        20   0  2276  1536  1440 S  0.0  0.0  0:00.00 /init
119 root        20   0  2292   100     0 S  0.0  0.0  0:00.00 /init
120 root        20   0  2292   108     0 S  0.0  0.0  0:00.00 /init
121 kizito      20   0  6072  5052  3332 S  0.0  0.1  0:00.08 -bash
132 kizito      20   0  5352  3676  2996 R  0.0  0.0  0:00.01 htop --no-color
```

Answer:

7-State process model:



Using the information in the table below, what is the parent process ID for the process that printed this table to the Linux shell?

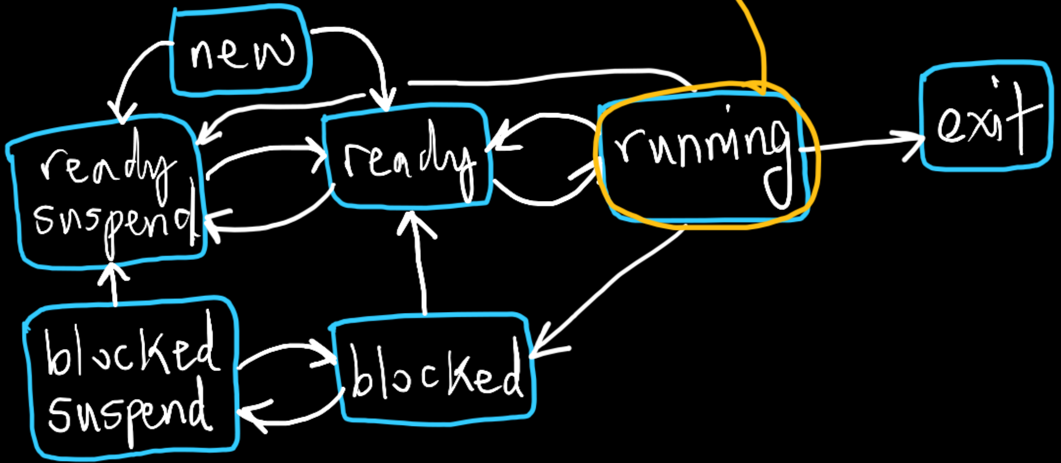


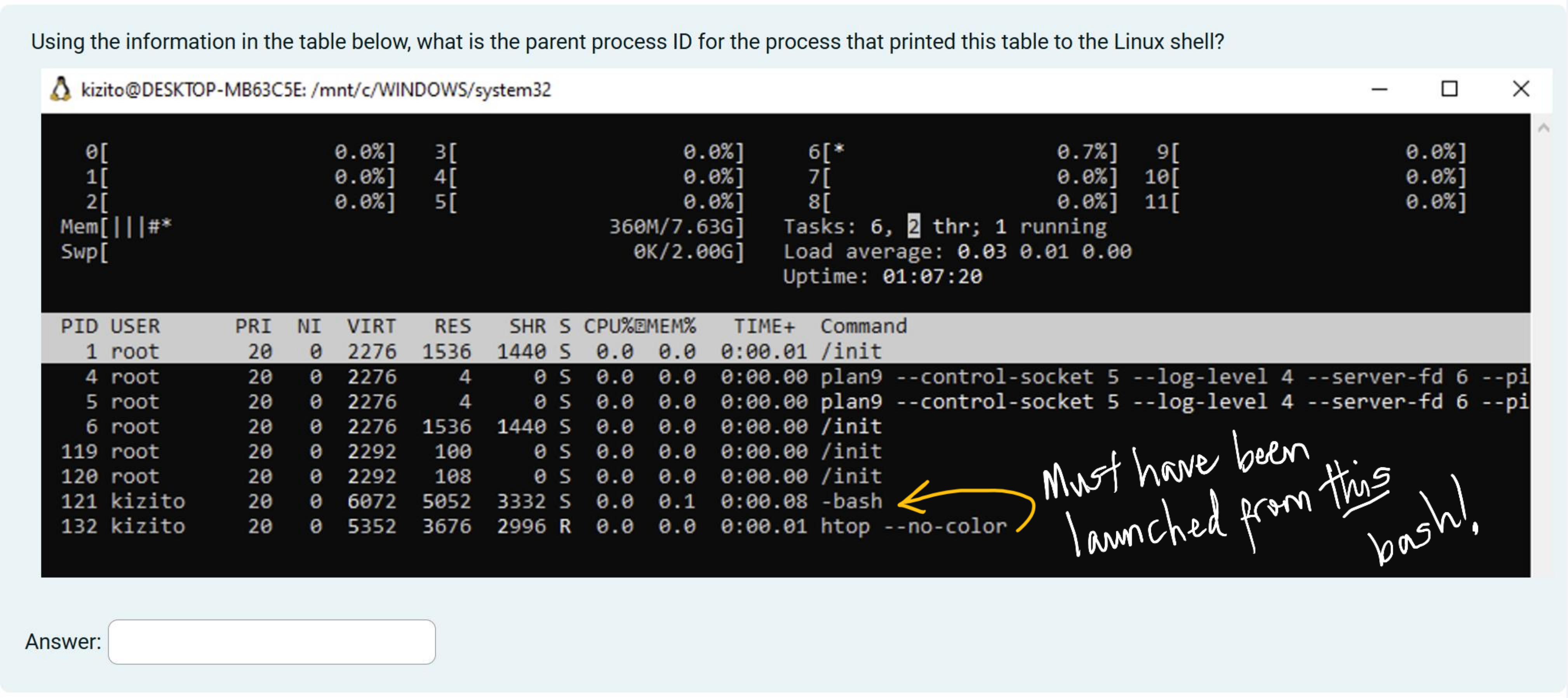
kizito@DESKTOP-MB63C5E: /mnt/c/WINDOWS/system32

```
  0[                   0.0%]   3[                    0.0%]    6[*                   0.7%]   9[                      0.0%]
  1[                   0.0%]   4[                    0.0%]    7[                    0.0%]  10[                      0.0%]
  2[                   0.0%]   5[                    0.0%]    8[                    0.0%]  11[                      0.0%]
Mem[|||#*                                   360M/7.63G]   Tasks: 6, 2 thr; 1 running
Swp[                                           0K/2.00G]   Load average: 0.03 0.01 0.00
                                                           Uptime: 01:07:20

PID USER       PRI  NI  VIRT   RES   SHR S CPU%▯MEM%   TIME+  Command
  1 root        20   0  2276  1536  1440 S  0.0  0.0  0:00.01 /init
  4 root        20   0  2276     4     0 S  0.0  0.0  0:00.00 plan9 --control-socket 5 --log-level 4 --server-fd 6 --pi
  5 root        20   0  2276     4     0 S  0.0  0.0  0:00.00 plan9 --control-socket 5 --log-level 4 --server-fd 6 --pi
  6 root        20   0  2276  1536  1440 S  0.0  0.0  0:00.00 /init
119 root        20   0  2292   100     0 S  0.0  0.0  0:00.00 /init
120 root        20   0  2292   108     0 S  0.0  0.0  0:00.00 /init
121 kizito      20   0  6072  5052  3332 S  0.0  0.1  0:00.08 -bash
132 kizito      20   0  5352  3676  2996 R  0.0  0.0  0:00.01 htop --no-color
```

Answer:

7-State process model:





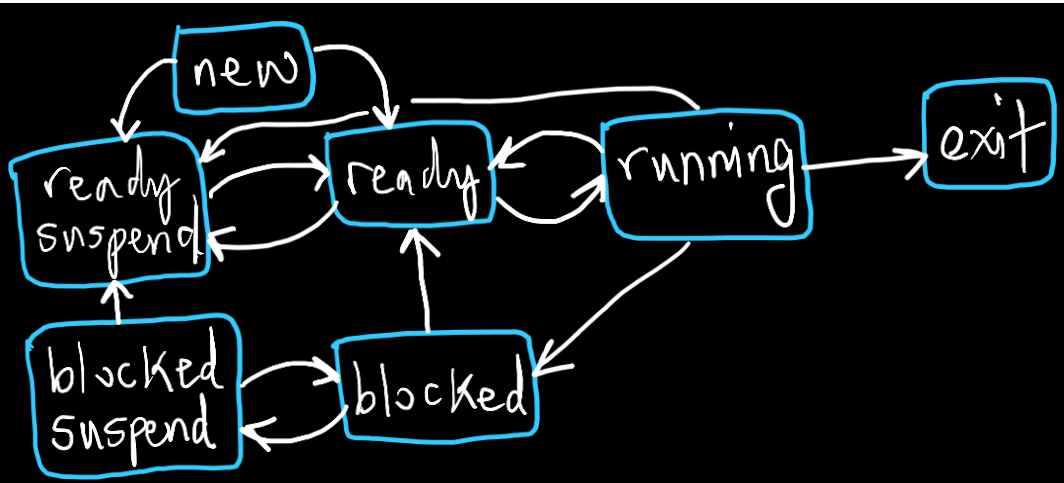
Using the information in the table below, what is the parent process ID for the process that printed this table to the Linux shell?

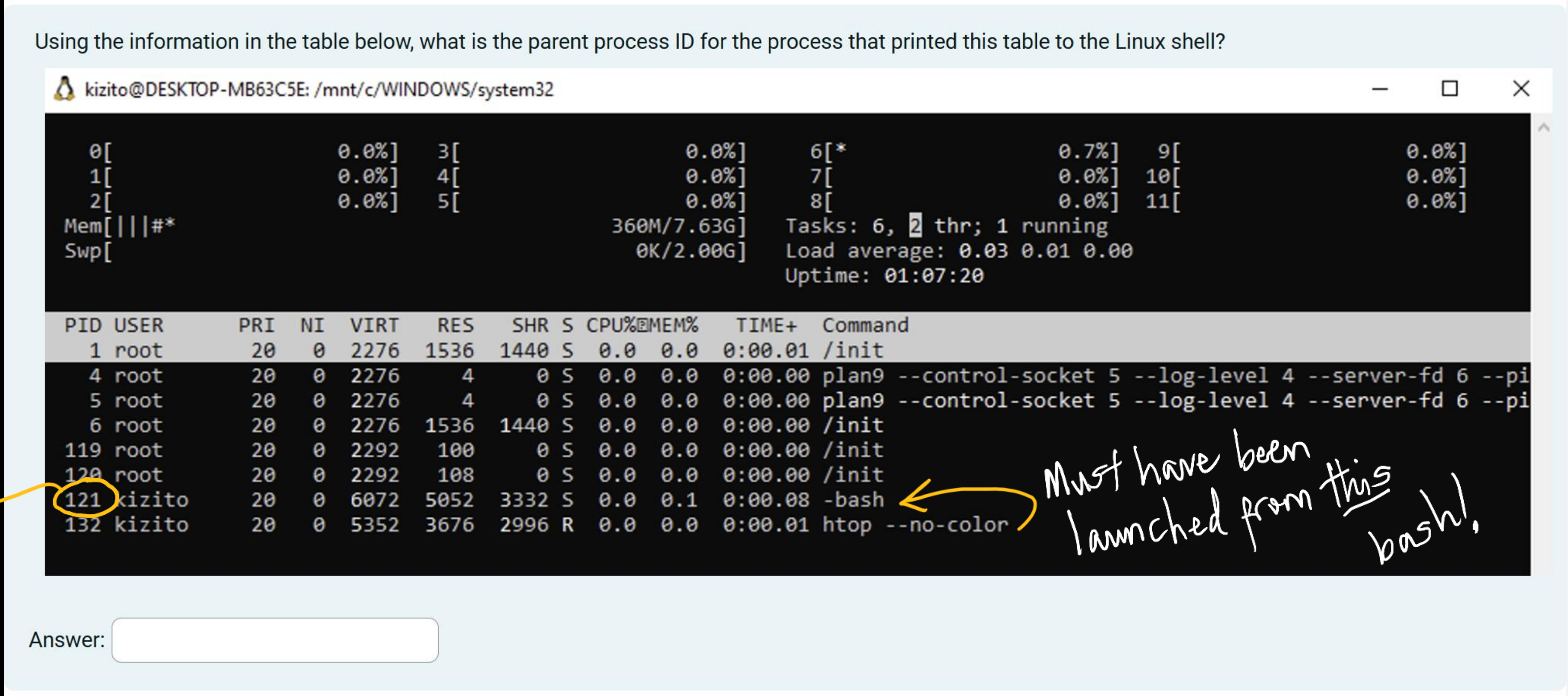
```
  0[                    0.0%]   3[                    0.0%]    6[*                   0.7%]   9[                    0.0%]
  1[                    0.0%]   4[                    0.0%]    7[                    0.0%]  10[                    0.0%]
  2[                    0.0%]   5[                    0.0%]    8[                    0.0%]  11[                    0.0%]
Mem[|||#*                                     360M/7.63G]  Tasks: 6, 2 thr; 1 running
Swp[                                            0K/2.00G]  Load average: 0.03 0.01 0.00
                                                           Uptime: 01:07:20

PID USER       PRI  NI  VIRT   RES   SHR S CPU%▯MEM%   TIME+  Command
  1 root        20   0  2276  1536  1440 S  0.0  0.0  0:00.01 /init
  4 root        20   0  2276     4     0 S  0.0  0.0  0:00.00 plan9 --control-socket 5 --log-level 4 --server-fd 6 --pi
  5 root        20   0  2276     4     0 S  0.0  0.0  0:00.00 plan9 --control-socket 5 --log-level 4 --server-fd 6 --pi
  6 root        20   0  2276  1536  1440 S  0.0  0.0  0:00.00 /init
119 root        20   0  2292   100     0 S  0.0  0.0  0:00.00 /init
120 root        20   0  2292   108     0 S  0.0  0.0  0:00.00 /init
121 kizito      20   0  6072  5052  3332 S  0.0  0.1  0:00.08 -bash
132 kizito      20   0  5352  3676  2996 R  0.0  0.0  0:00.01 htop --no-color
```

Must have been launched from this bash!

Answer:

7-State process model:





Using the information in the table below, what is the parent process ID for the process that printed this table to the Linux shell?

kizito@DESKTOP-MB63C5E: /mnt/c/WINDOWS/system32

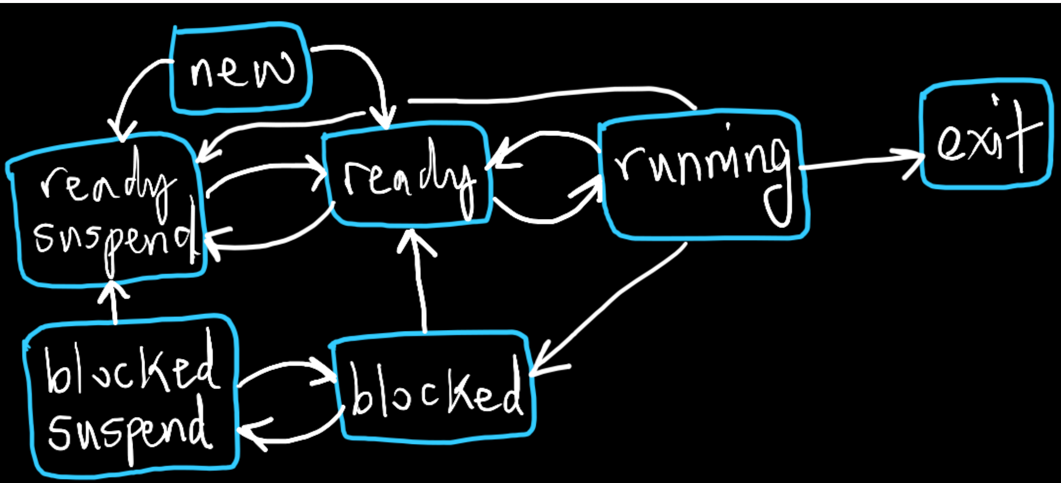
```
  0[                    0.0%]   3[                        0.0%]    6[*                       0.7%]   9[                        0.0%]
  1[                    0.0%]   4[                        0.0%]    7[                        0.0%]  10[                        0.0%]
  2[                    0.0%]   5[                        0.0%]    8[                        0.0%]  11[                        0.0%]
Mem[|||#*                                        360M/7.63G]   Tasks: 6, 2 thr; 1 running
Swp[                                               0K/2.00G]   Load average: 0.03 0.01 0.00
                                                               Uptime: 01:07:20

PID USER       PRI  NI  VIRT   RES   SHR S CPU%▯MEM%   TIME+  Command
  1 root        20   0  2276  1536  1440 S  0.0  0.0  0:00.01 /init
  4 root        20   0  2276     4     0 S  0.0  0.0  0:00.00 plan9 --control-socket 5 --log-level 4 --server-fd 6 --pi
  5 root        20   0  2276     4     0 S  0.0  0.0  0:00.00 plan9 --control-socket 5 --log-level 4 --server-fd 6 --pi
  6 root        20   0  2276  1536  1440 S  0.0  0.0  0:00.00 /init
119 root        20   0  2292   100     0 S  0.0  0.0  0:00.00 /init
120 root        20   0  2292   108     0 S  0.0  0.0  0:00.00 /init
121 kizito      20   0  6072  5052  3332 S  0.0  0.1  0:00.08 -bash
132 kizito      20   0  5352  3676  2996 R  0.0  0.0  0:00.01 htop --no-color
```

Answer:

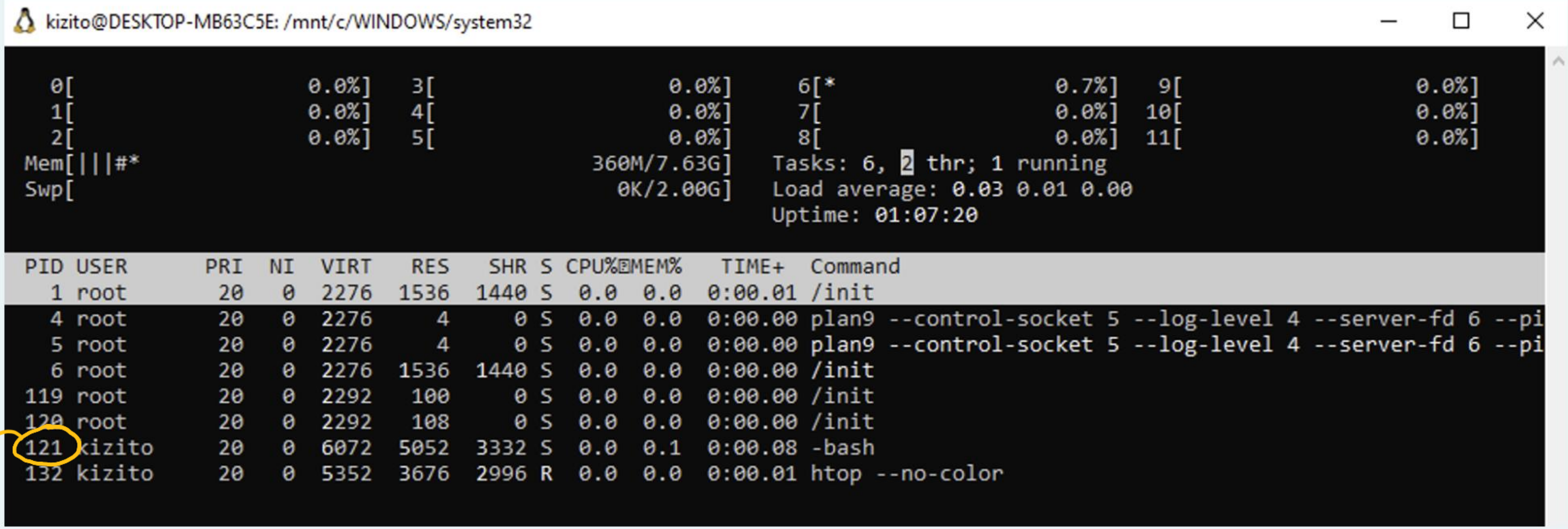
ID of bash process

Must have been launched from this bash!

7-State process model:



Using the information in the table below, what is the parent process ID for the process that printed this table to the Linux shell?

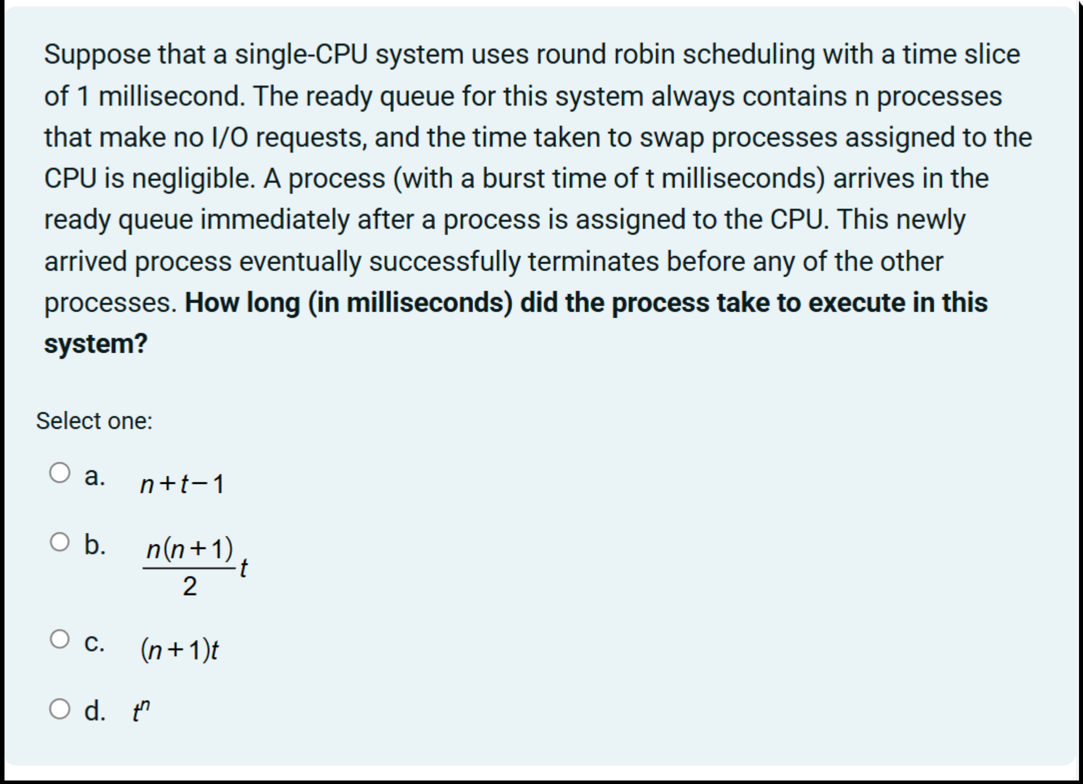


kizito@DESKTOP-MB63C5E: /mnt/c/WINDOWS/system32

```
  0[                    0.0%]   3[                    0.0%]    6[*                   0.7%]   9[                    0.0%]
  1[                    0.0%]   4[                    0.0%]    7[                    0.0%]  10[                    0.0%]
  2[                    0.0%]   5[                    0.0%]    8[                    0.0%]  11[                    0.0%]
Mem[|||#*                               360M/7.63G]   Tasks: 6, 2 thr; 1 running
Swp[                                      0K/2.00G]   Load average: 0.03 0.01 0.00
                                                      Uptime: 01:07:20

PID USER       PRI  NI  VIRT   RES   SHR S CPU%▯MEM%   TIME+  Command
  1 root        20   0  2276  1536  1440 S  0.0  0.0  0:00.01 /init
  4 root        20   0  2276     4     0 S  0.0  0.0  0:00.00 plan9 --control-socket 5 --log-level 4 --server-fd 6 --pi
  5 root        20   0  2276     4     0 S  0.0  0.0  0:00.00 plan9 --control-socket 5 --log-level 4 --server-fd 6 --pi
  6 root        20   0  2276  1536  1440 S  0.0  0.0  0:00.00 /init
119 root        20   0  2292   100     0 S  0.0  0.0  0:00.00 /init
120 root        20   0  2292   108     0 S  0.0  0.0  0:00.00 /init
121 kizito      20   0  6072  5052  3332 S  0.0  0.1  0:00.08 -bash
132 kizito      20   0  5352  3676  2996 R  0.0  0.0  0:00.01 htop --no-color
```

ID of bash process

Answer: 121



Suppose that a single-CPU system uses round robin scheduling with a time slice of 1 millisecond. The ready queue for this system always contains n processes that make no I/O requests, and the time taken to swap processes assigned to the CPU is negligible. A process (with a burst time of t milliseconds) arrives in the ready queue immediately after a process is assigned to the CPU. This newly arrived process eventually successfully terminates before any of the other processes. **How long (in milliseconds) did the process take to execute in this system?**

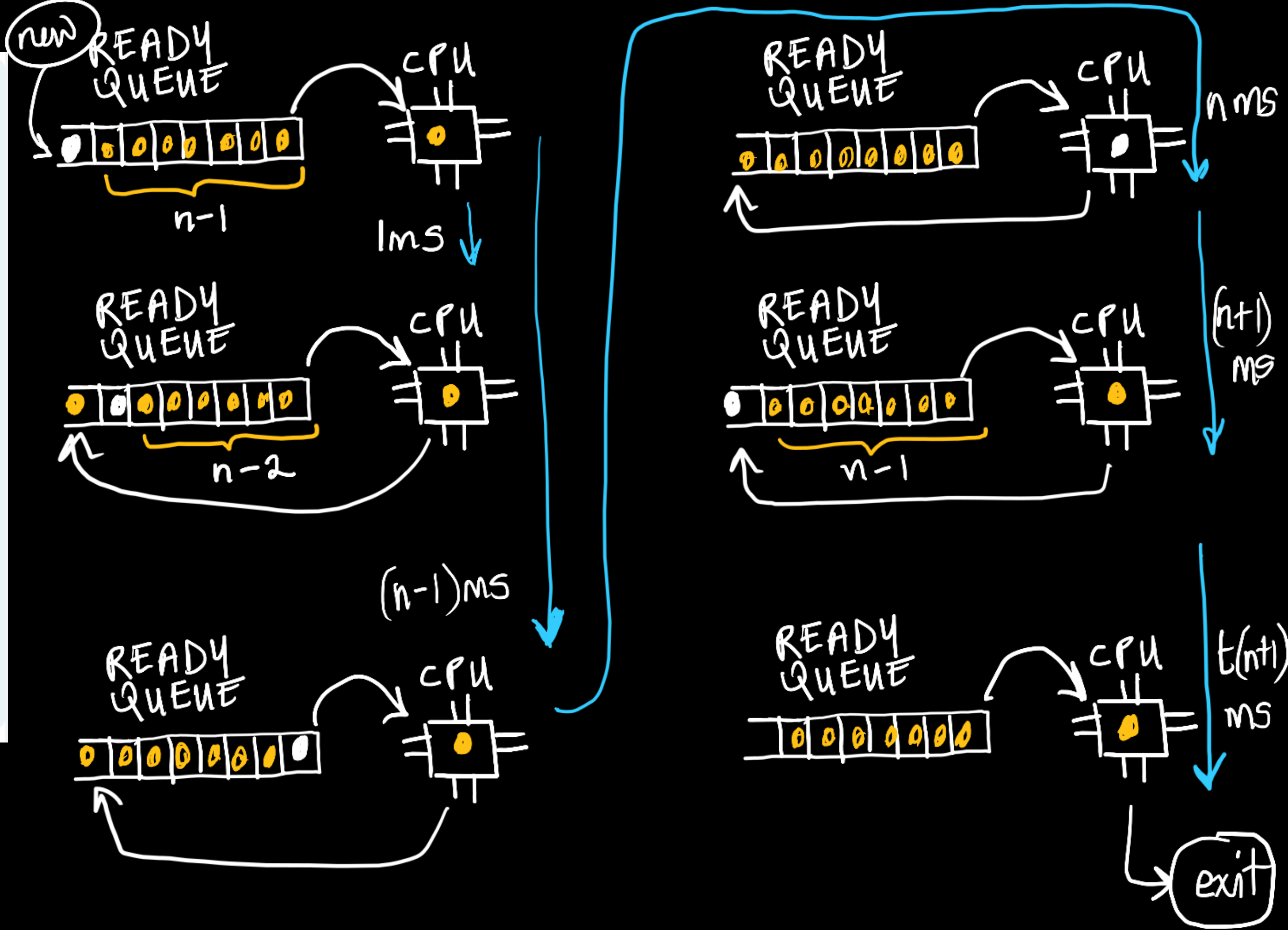
Select one:

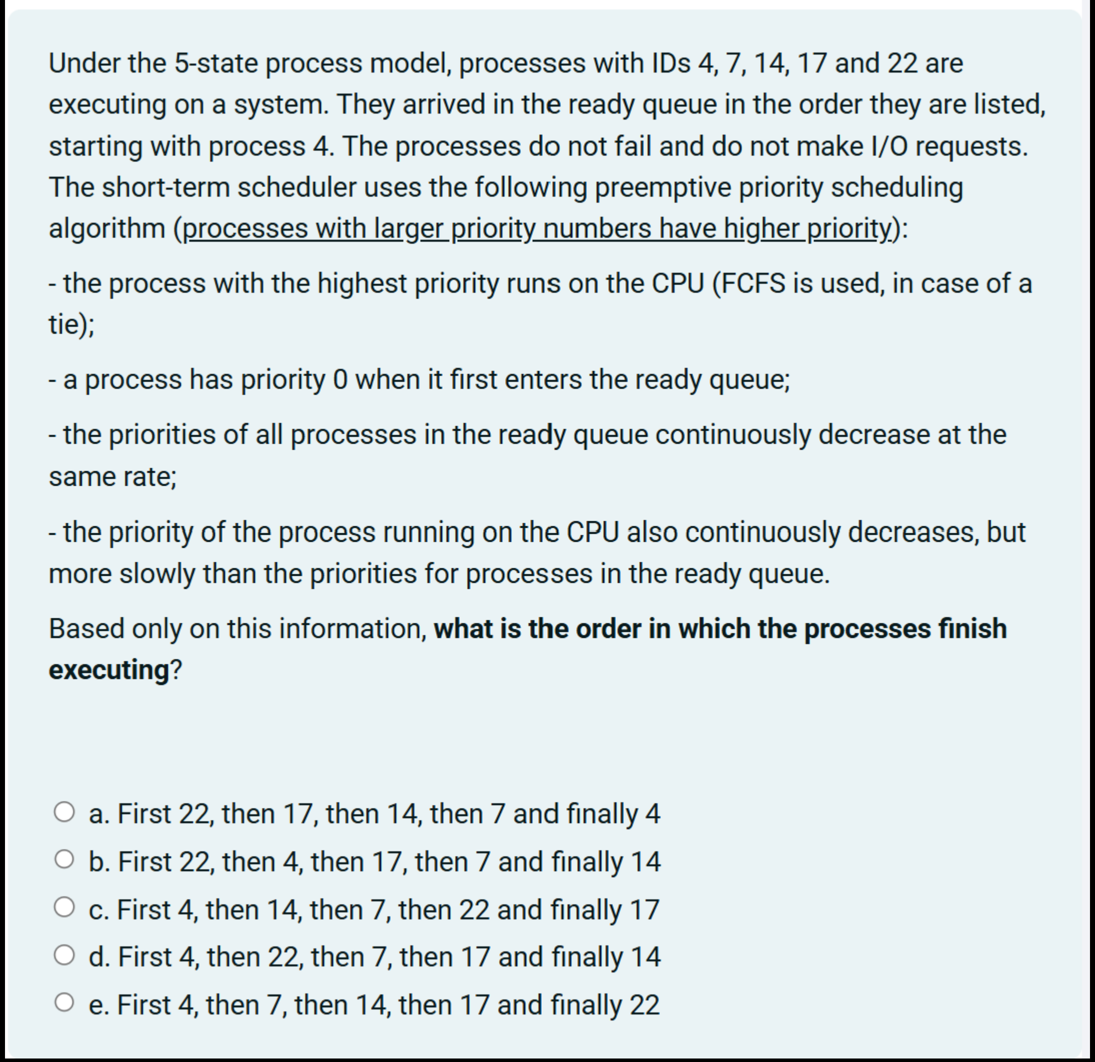
○ a. $n+t-1$

○ b. $\frac{n(n+1)}{2}t$

○ c. $(n+1)t$

○ d. t^n





Under the 5-state process model, processes with IDs 4, 7, 14, 17 and 22 are executing on a system. They arrived in the ready queue in the order they are listed, starting with process 4. The processes do not fail and do not make I/O requests. The short-term scheduler uses the following preemptive priority scheduling algorithm (processes with larger priority numbers have higher priority):

- the process with the highest priority runs on the CPU (FCFS is used, in case of a tie);

- a process has priority 0 when it first enters the ready queue;

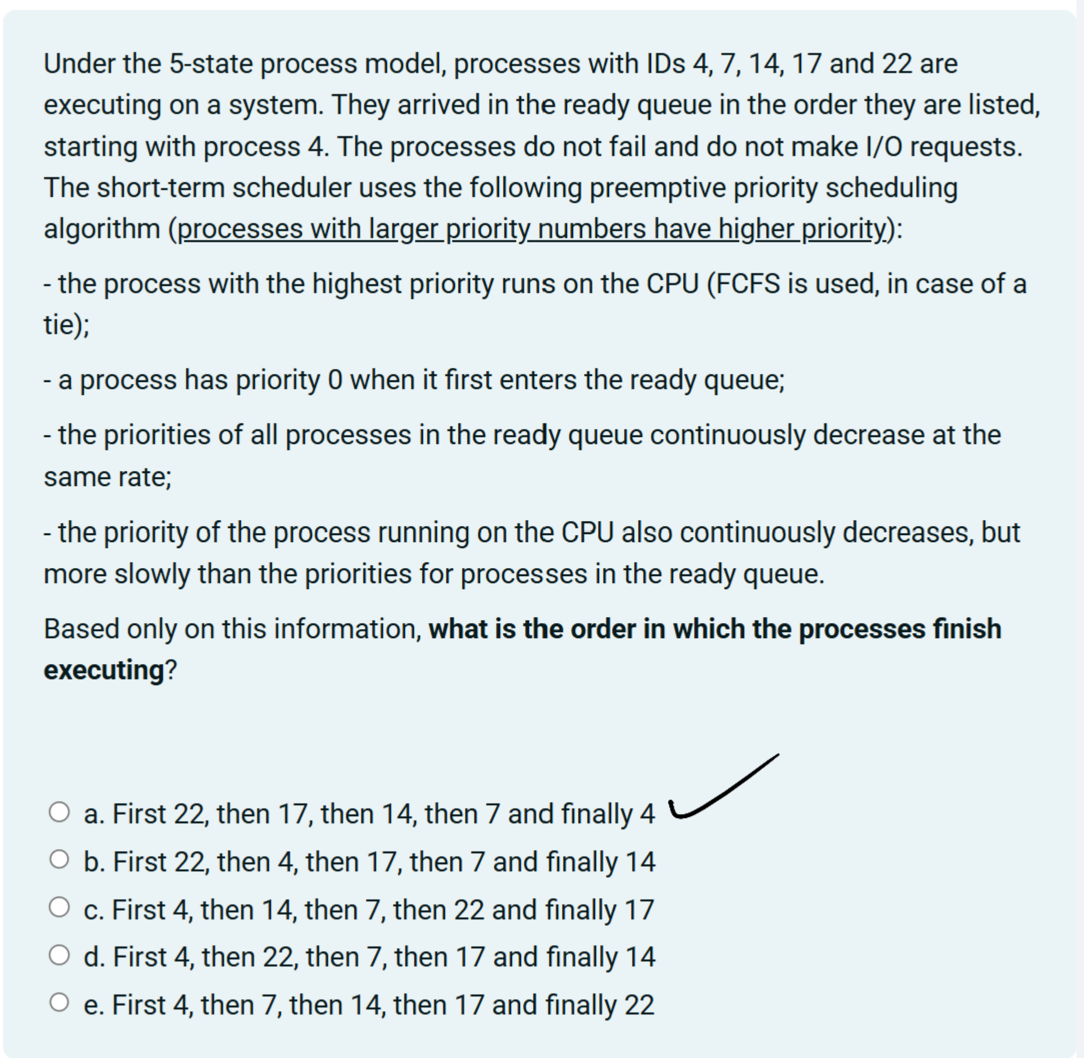
- the priorities of all processes in the ready queue continuously decrease at the same rate;

- the priority of the process running on the CPU also continuously decreases, but more slowly than the priorities for processes in the ready queue.

Based only on this information, **what is the order in which the processes finish executing**?

○ a. First 22, then 17, then 14, then 7 and finally 4
○ b. First 22, then 4, then 17, then 7 and finally 14
○ c. First 4, then 14, then 7, then 22 and finally 17
○ d. First 4, then 22, then 7, then 17 and finally 14
○ e. First 4, then 7, then 14, then 17 and finally 22

5-state process model:



Under the 5-state process model, processes with IDs 4, 7, 14, 17 and 22 are executing on a system. They arrived in the ready queue in the order they are listed, starting with process 4. The processes do not fail and do not make I/O requests. The short-term scheduler uses the following preemptive priority scheduling algorithm (processes with larger priority numbers have higher priority):

- the process with the highest priority runs on the CPU (FCFS is used, in case of a tie);

- a process has priority 0 when it first enters the ready queue;

- the priorities of all processes in the ready queue continuously decrease at the same rate;

- the priority of the process running on the CPU also continuously decreases, but more slowly than the priorities for processes in the ready queue.

Based only on this information, **what is the order in which the processes finish executing**?

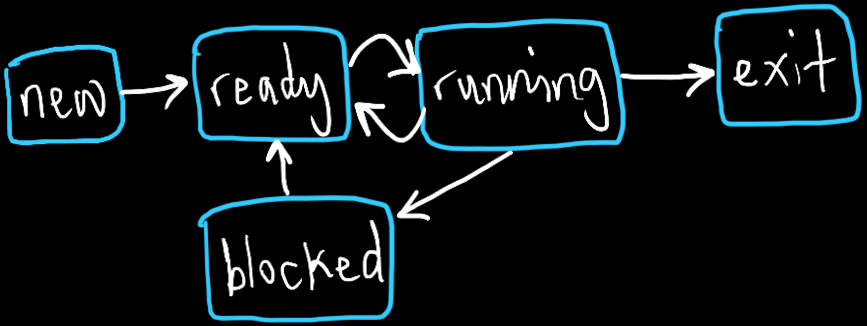
○ [✓ a. First 22, then 17, then 14, then 7 and finally 4]

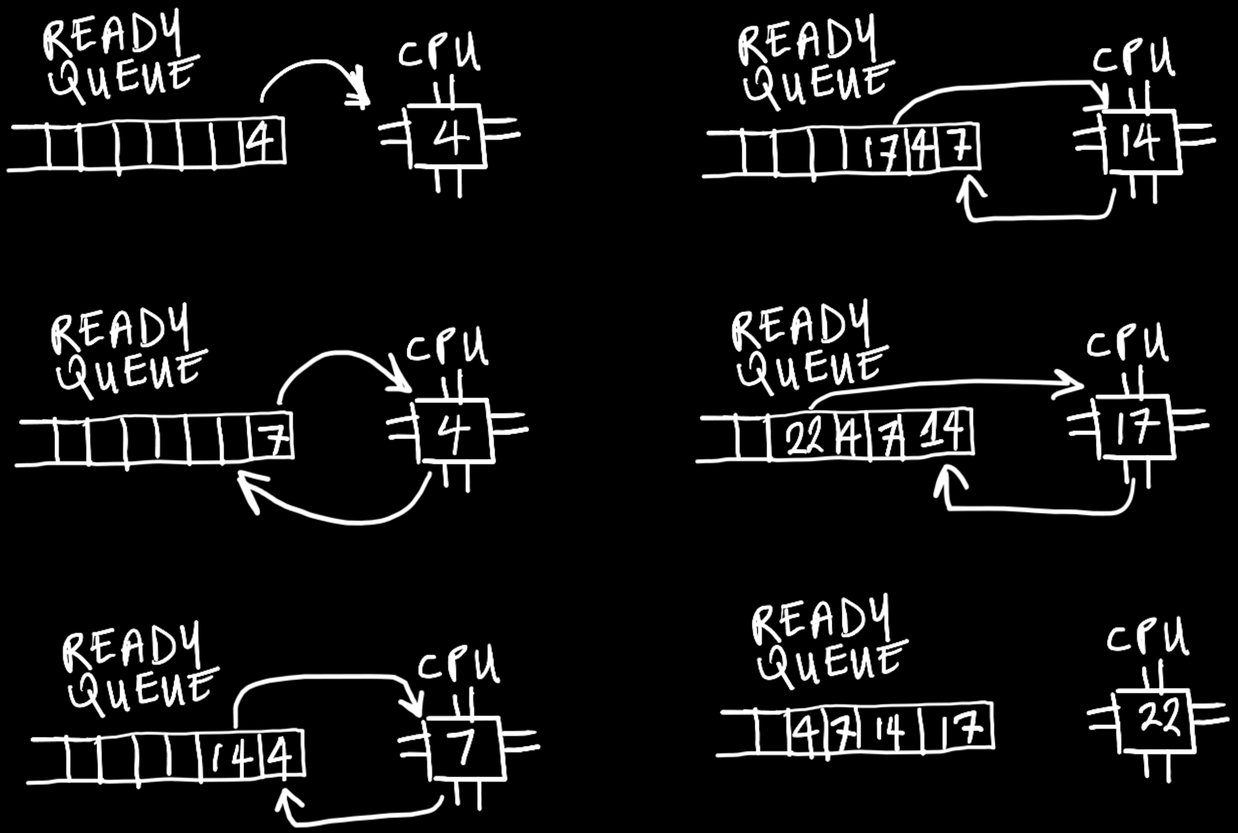
○ b. First 22, then 4, then 17, then 7 and finally 14

○ c. First 4, then 14, then 7, then 22 and finally 17

○ d. First 4, then 22, then 7, then 17 and finally 14

○ e. First 4, then 7, then 14, then 17 and finally 22


new
ready
running
exit
blocked


READY QUEUE
4
CPU
4
READY QUEUE
17 4 7
CPU
14
READY QUEUE
7
CPU
4
READY QUEUE
22 4 7 14
CPU
17
READY QUEUE
14 4
CPU
7
READY QUEUE
4 7 14 17
CPU
22