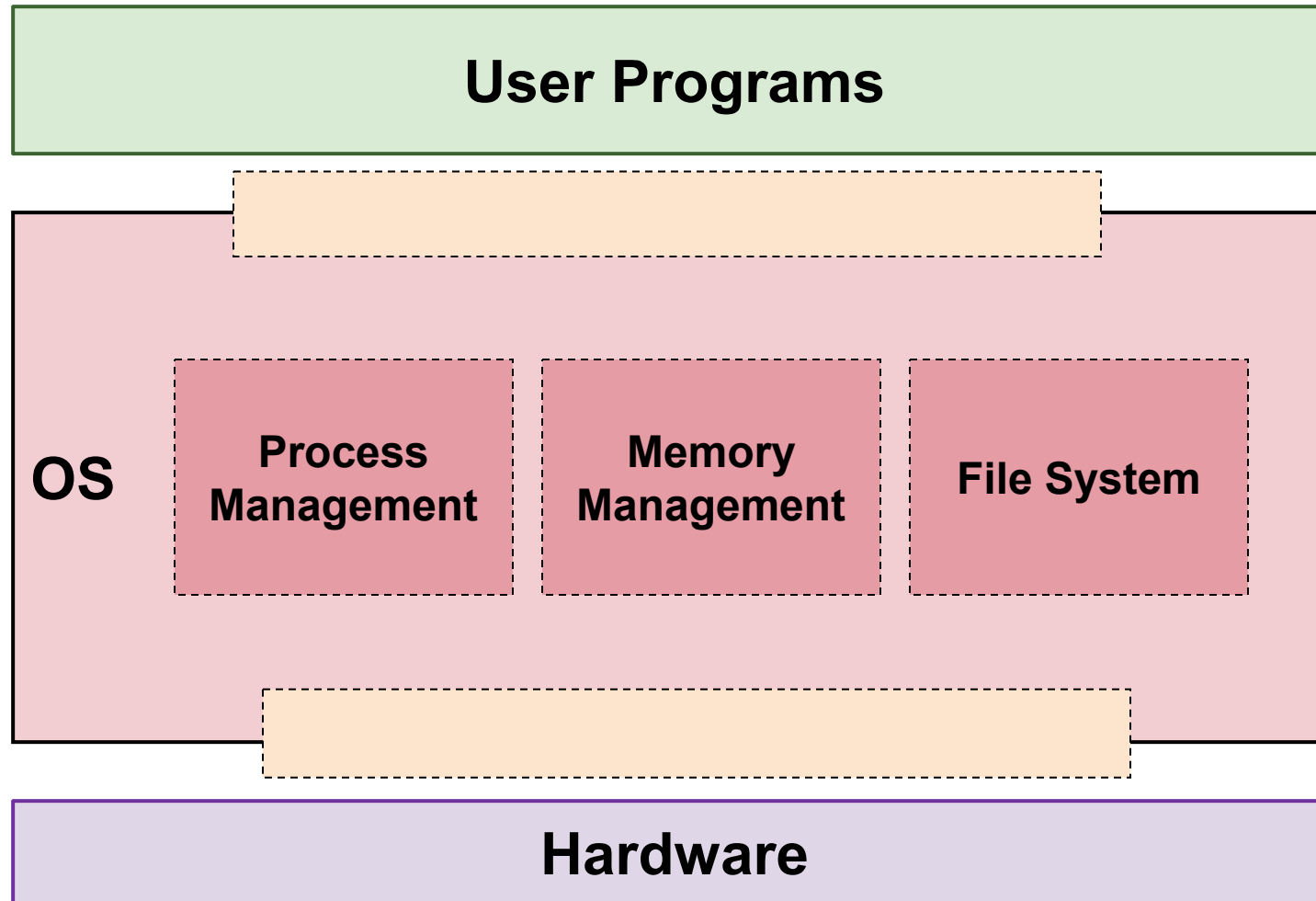
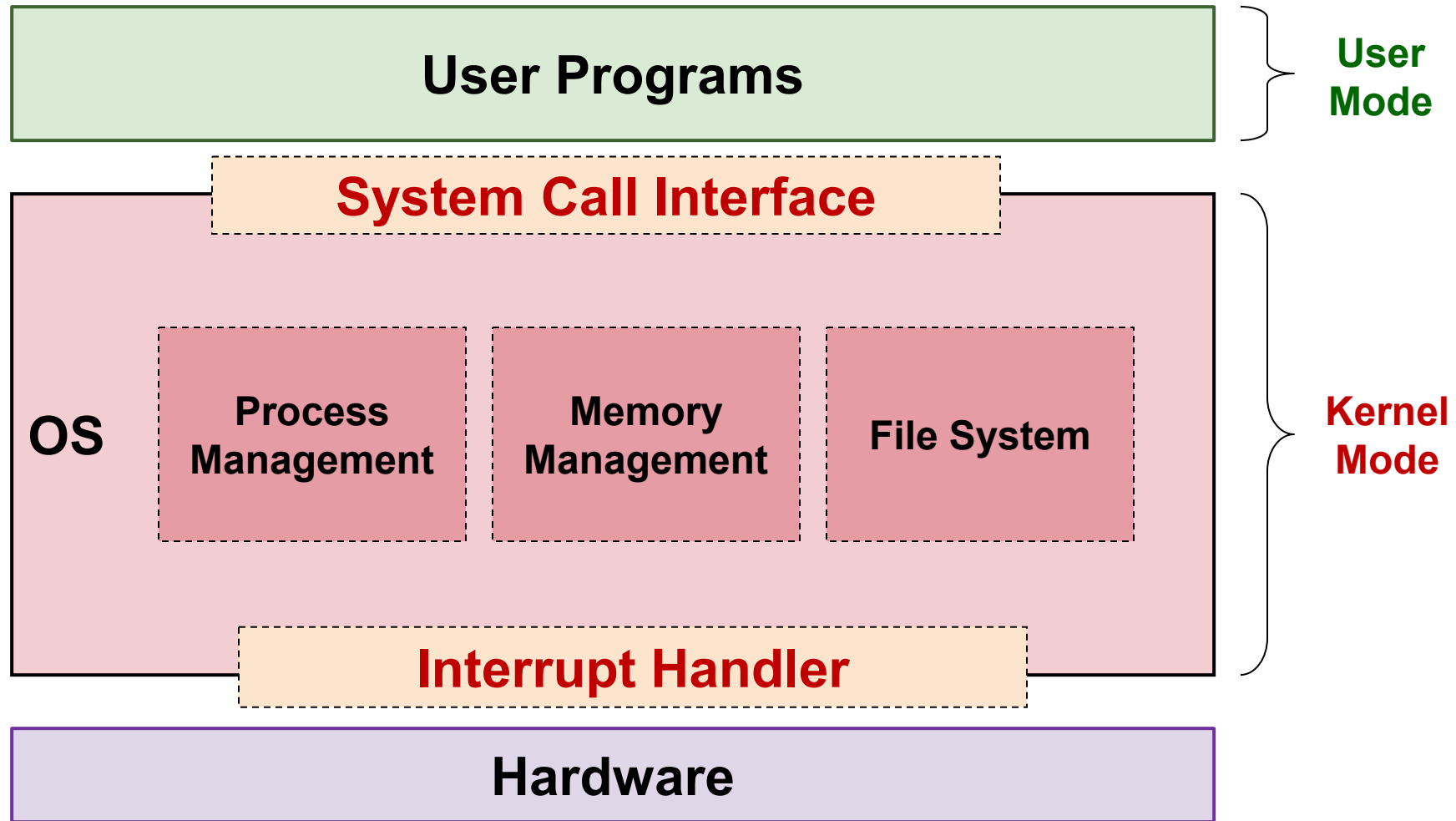

Big Bang & Revision

Lecture 12 (**Live** Version)

OS Structure (Monolithic)



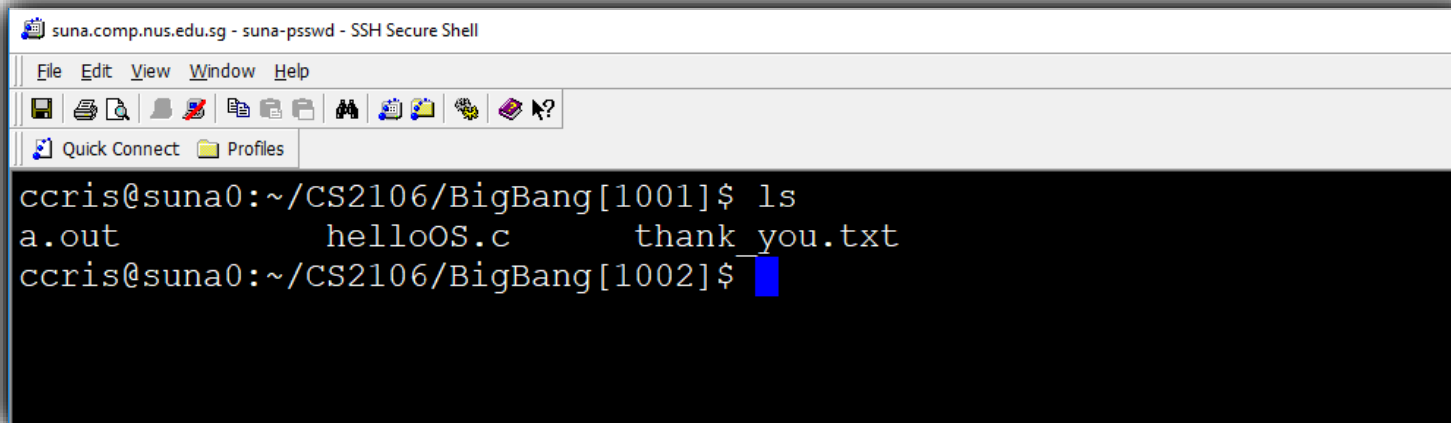
OS Structure (Monolithic)



*23 + 11 + **lots of** hours just to learn about what happened in **1 second***

THE STORY OF SHELL INTERPRETER

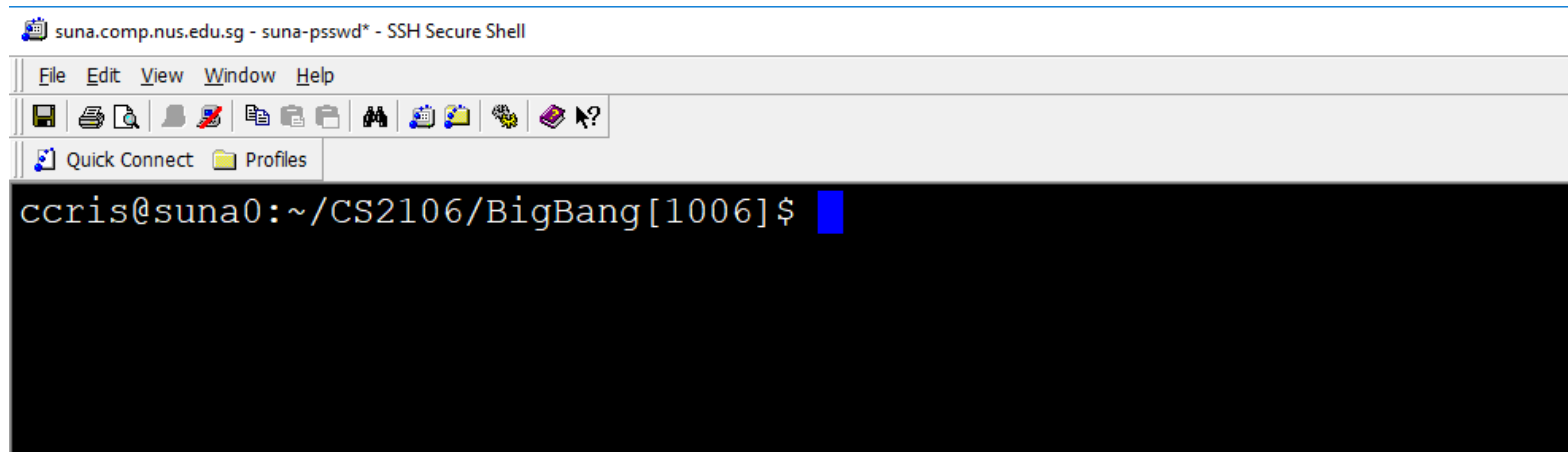
Ah.... Ah.... so simple..... **or is it?**



The image shows a screenshot of an SSH terminal window. The title bar reads "suna.comp.nus.edu.sg - suna-psswd - SSH Secure Shell". Below the title bar is a menu bar with "File", "Edit", "View", "Window", and "Help". Underneath the menu bar is a toolbar with various icons for file operations like save, print, and search. Below the toolbar is a tab bar with "Quick Connect" and "Profiles". The main area of the terminal is black with white text. It shows a user named "ccris" at a host named "suna0" in the directory "~/CS2106/BigBang". The user has entered the command "ls" and the output is displayed: "a.out", "helloOS.c", and "thank_you.txt". The prompt for the next command is shown as "ccris@suna0:~/CS2106/BigBang[1002]\$" with a blue cursor.

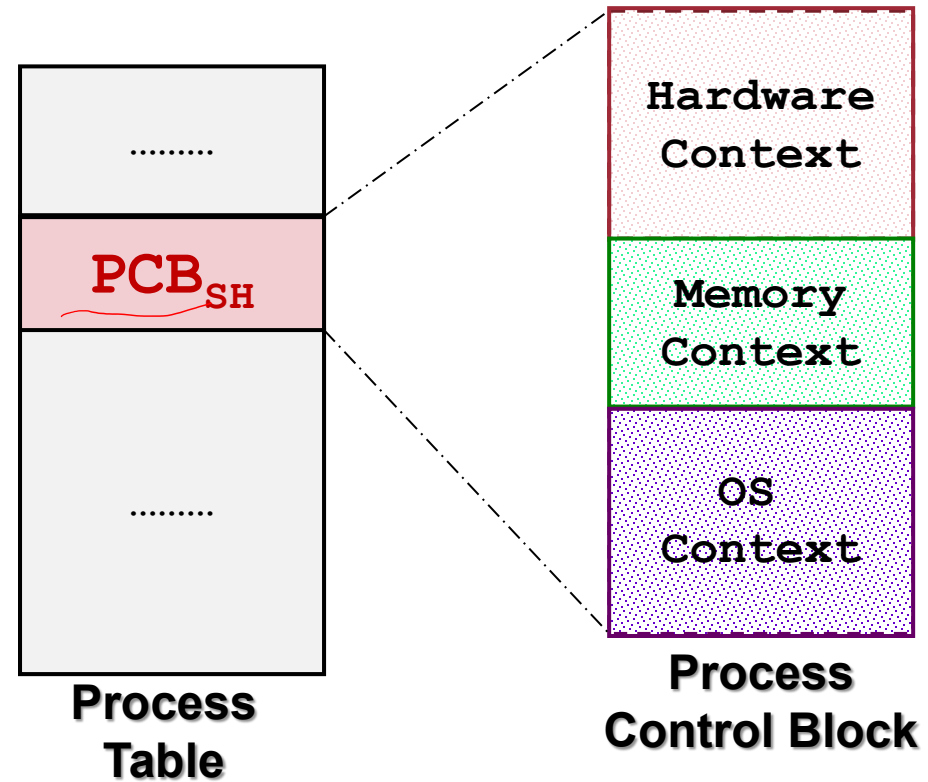
```
suna.comp.nus.edu.sg - suna-psswd - SSH Secure Shell
File Edit View Window Help
[Toolbar icons]
Quick Connect Profiles
ccris@suna0:~/CS2106/BigBang[1001]$ ls
a.out      helloOS.c  thank_you.txt
ccris@suna0:~/CS2106/BigBang[1002]$
```

Here we *go*

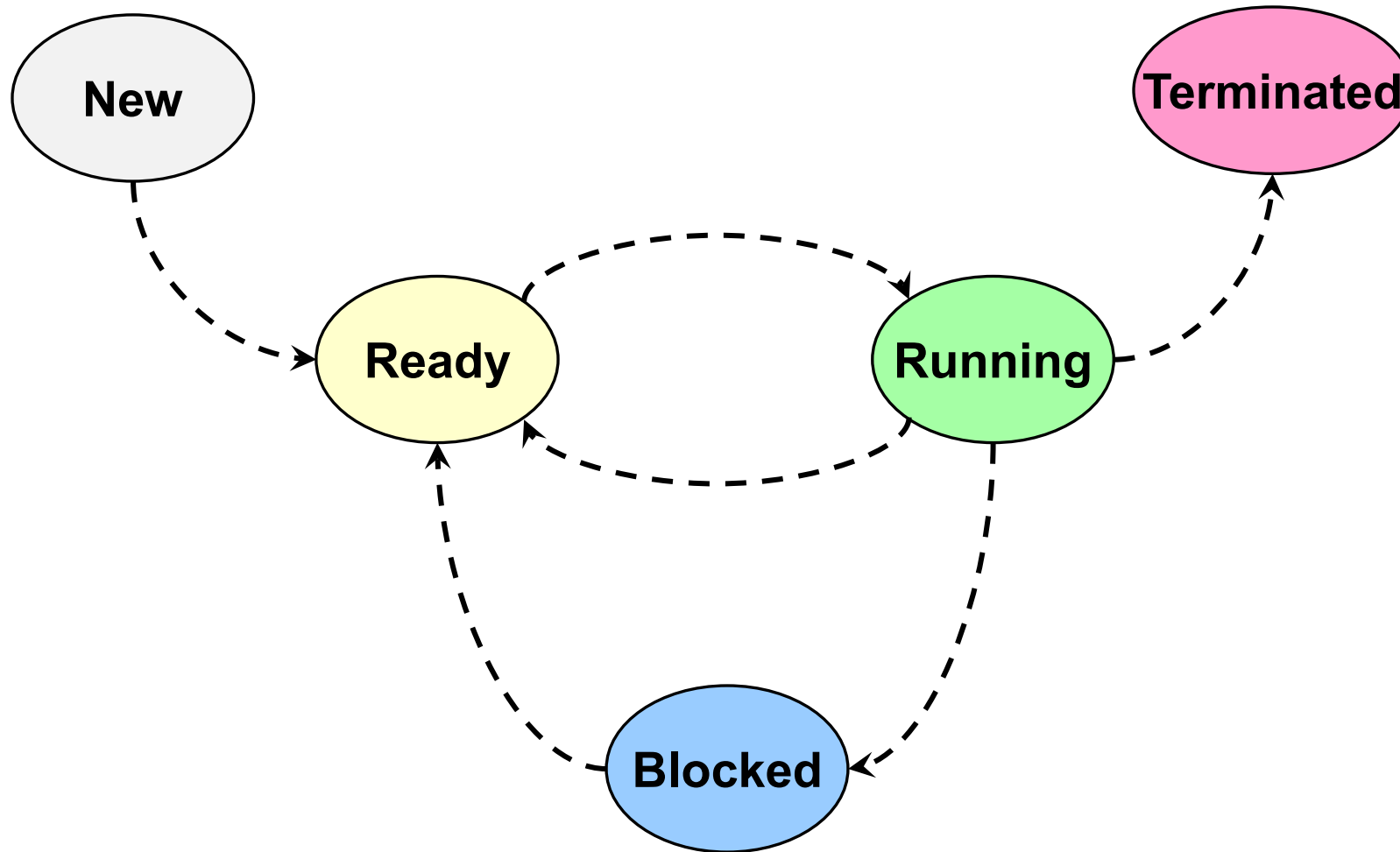


```
suna.comp.nus.edu.sg - suna-psswd* - SSH Secure Shell
File Edit View Window Help
[Toolbar icons]
Quick Connect Profiles
ccris@suna0:~/CS2106/BigBang[1006]$
```

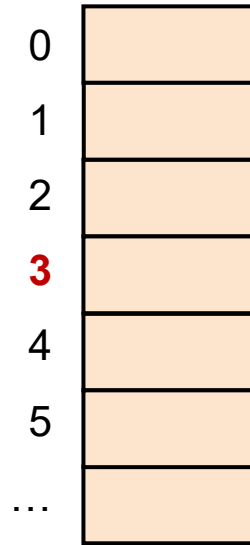
Process Table & Process Control Block



The Interpreter is in.....



User press "1"



Keyboard Handler

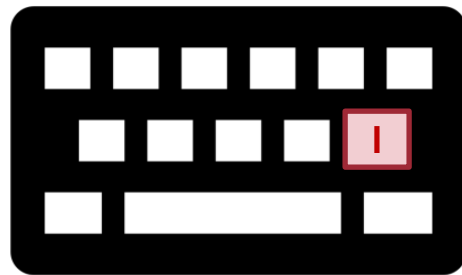
$\$r1 \leftarrow \text{key pressed}$

$\text{mem}[\$r2] \leftarrow \$r1$

Done!

Interrupt Handler

3

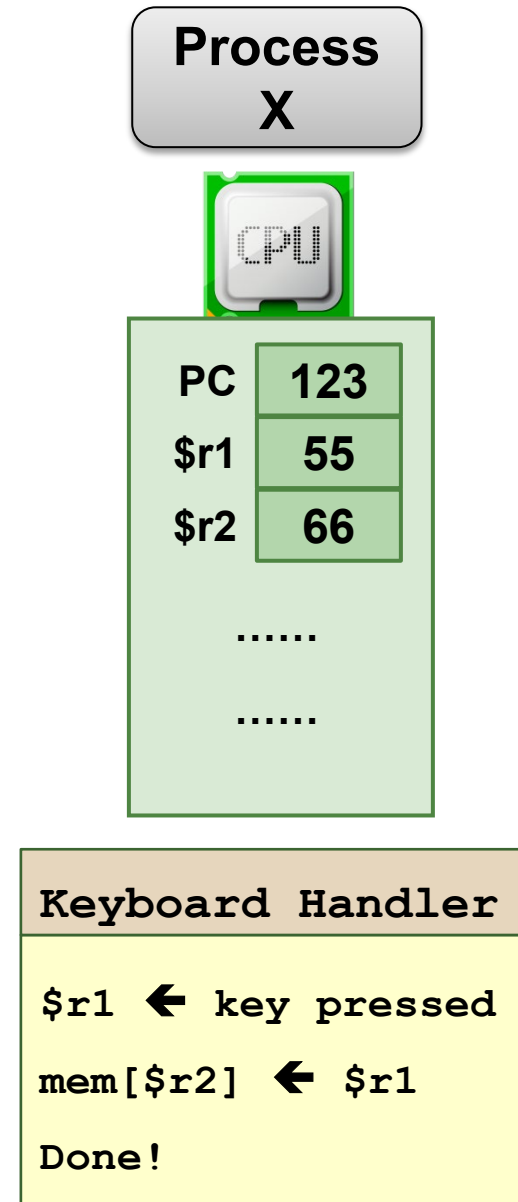
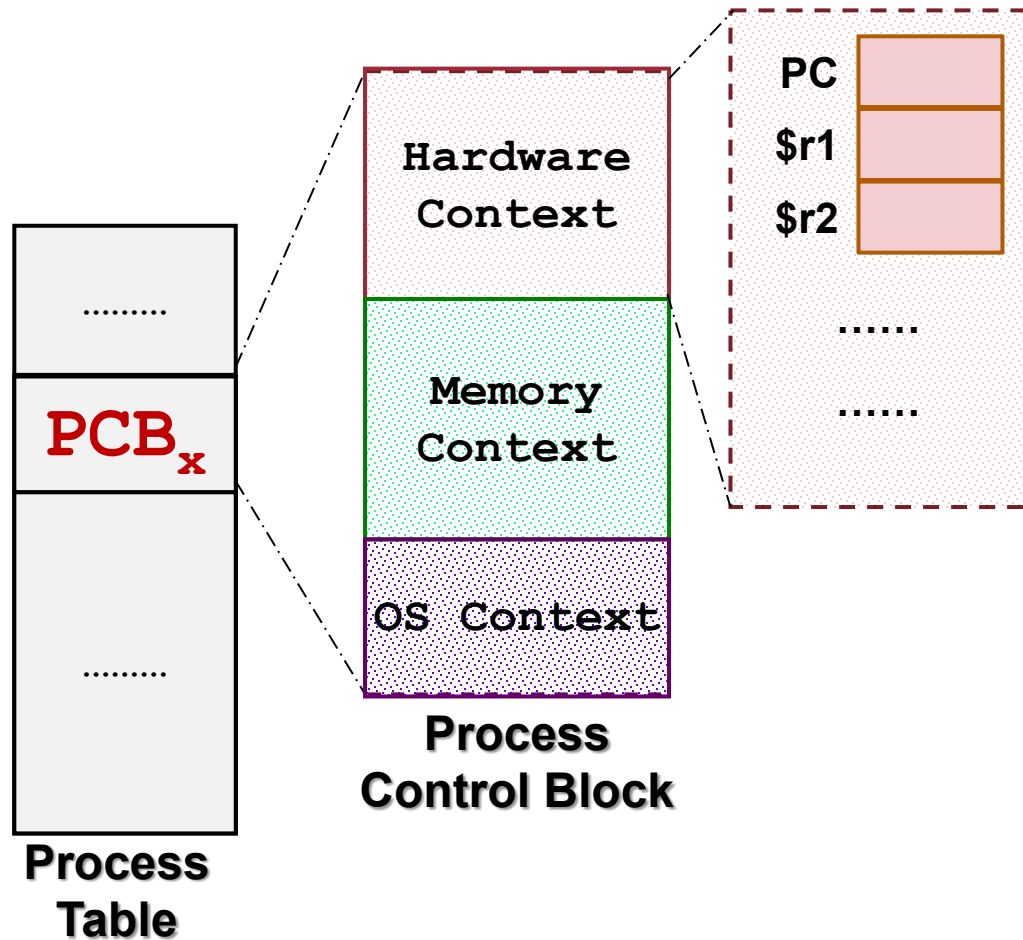


Process X:
Hey! ☹️



PC	123
$\$r1$	55
$\$r2$	66
.....	
.....	

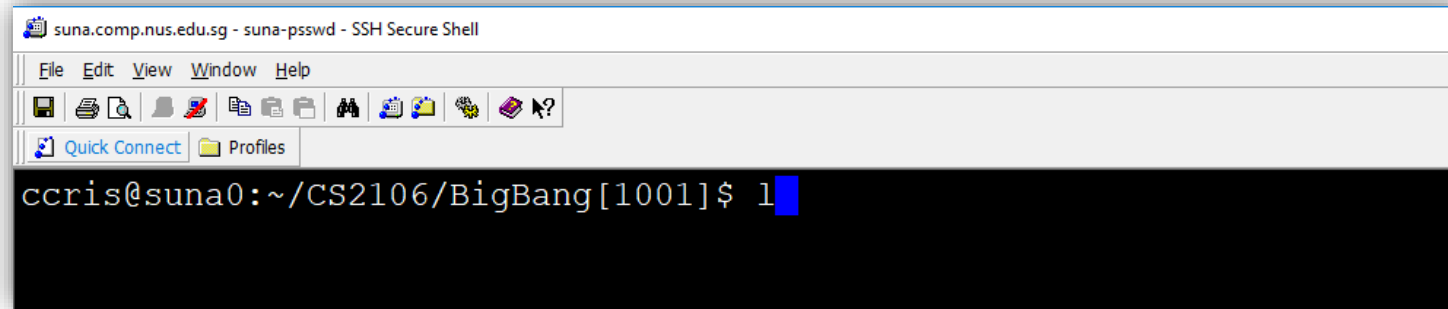
Sorry to **interrupt** you....



Interrupt steps

- Give the sequence of steps for handling an interrupt.
 - ❑ Interrupt occurs
 - ❑ Save registers/CPU state
 - ❑ Perform the handler routine
 - ❑ Restore registers/CPU state
 - ❑ Return from interrupt

Rinse and Repeat.....

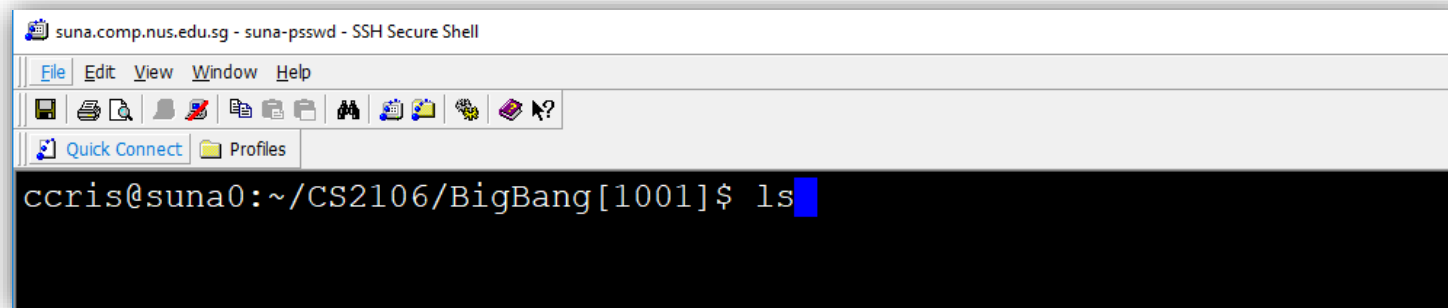


sunu.comp.nus.edu.sg - sunu-psswd - SSH Secure Shell

File Edit View Window Help

Quick Connect Profiles

```
ccris@sunu0:~/CS2106/BigBang[1001]$ l
```

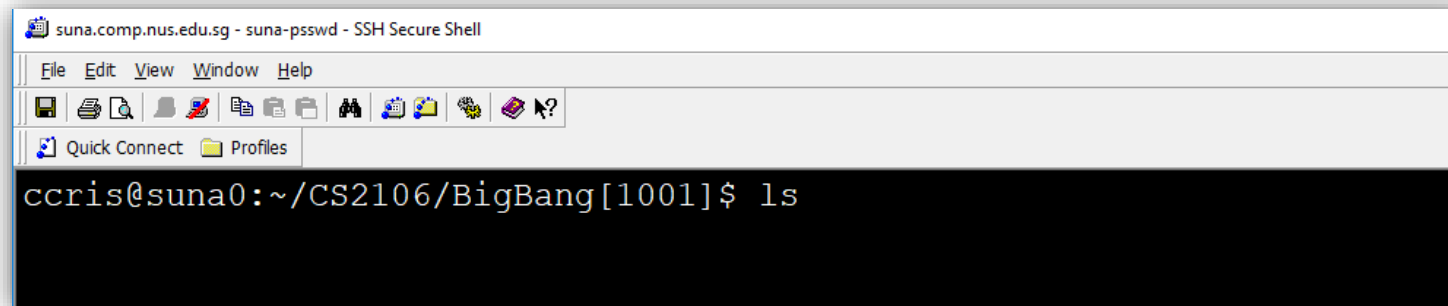


sunu.comp.nus.edu.sg - sunu-psswd - SSH Secure Shell

File Edit View Window Help

Quick Connect Profiles

```
ccris@sunu0:~/CS2106/BigBang[1001]$ ls
```



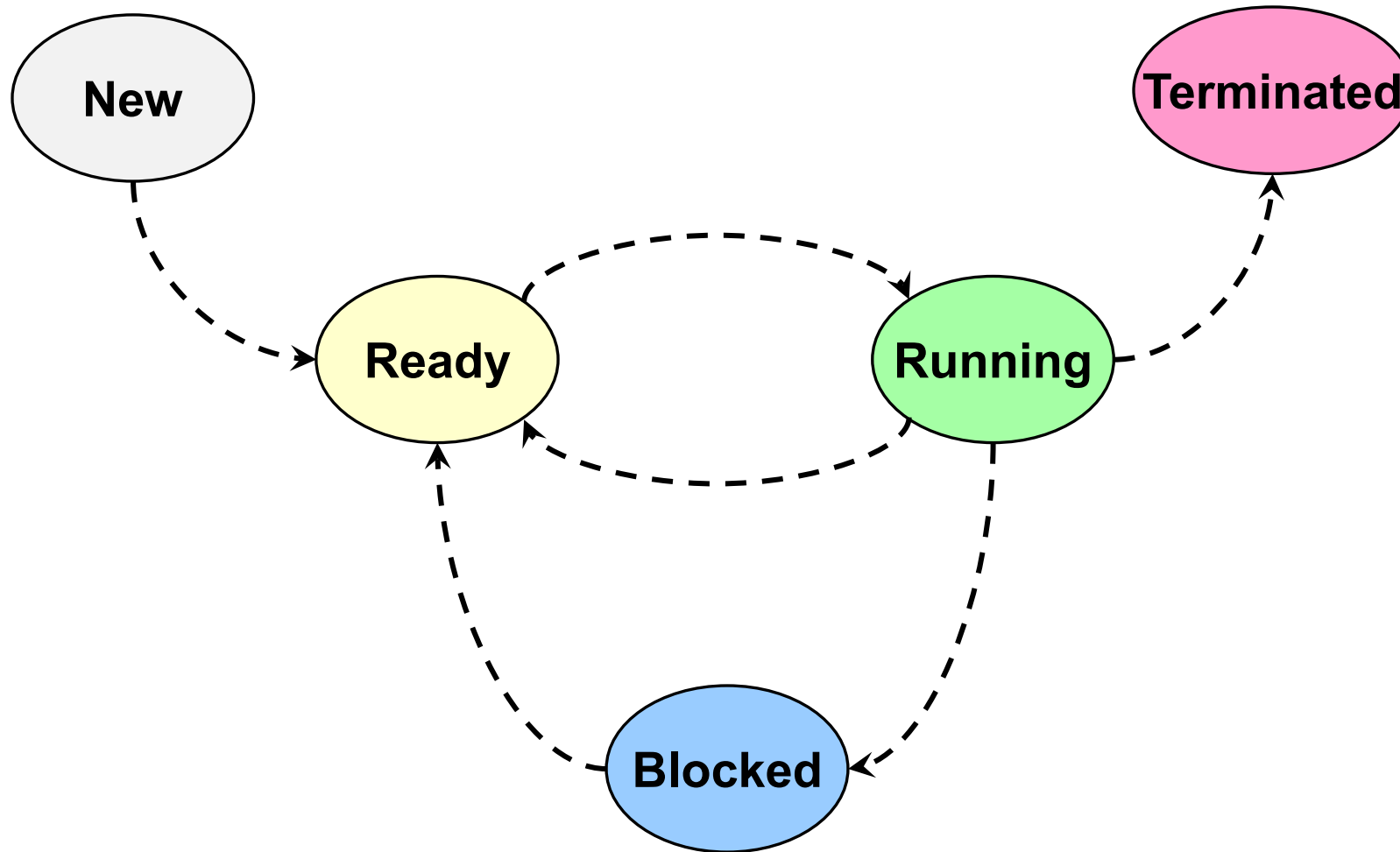
sunu.comp.nus.edu.sg - sunu-psswd - SSH Secure Shell

File Edit View Window Help

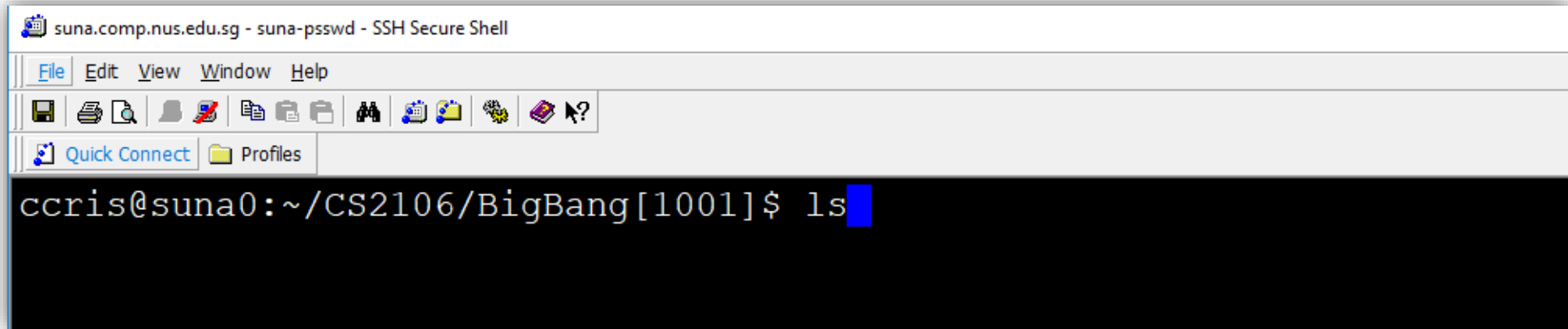
Quick Connect Profiles

```
ccris@sunu0:~/CS2106/BigBang[1001]$ ls
```

The **interpreter** is now...



User entered "ls", the interpreter will...



The screenshot shows a terminal window titled "suna.comp.nus.edu.sg - suna-psswd - SSH Secure Shell". The window has a menu bar with "File", "Edit", "View", "Window", and "Help". Below the menu bar is a toolbar with various icons. The terminal text shows the prompt "ccris@suna0:~/CS2106/BigBang[1001]" followed by the command "ls" which is currently being typed, with a blue cursor at the end of the command.

Typical Steps for Shell Interpreter

```
UserCmd ← read from keyboard
```

```
fork()
```



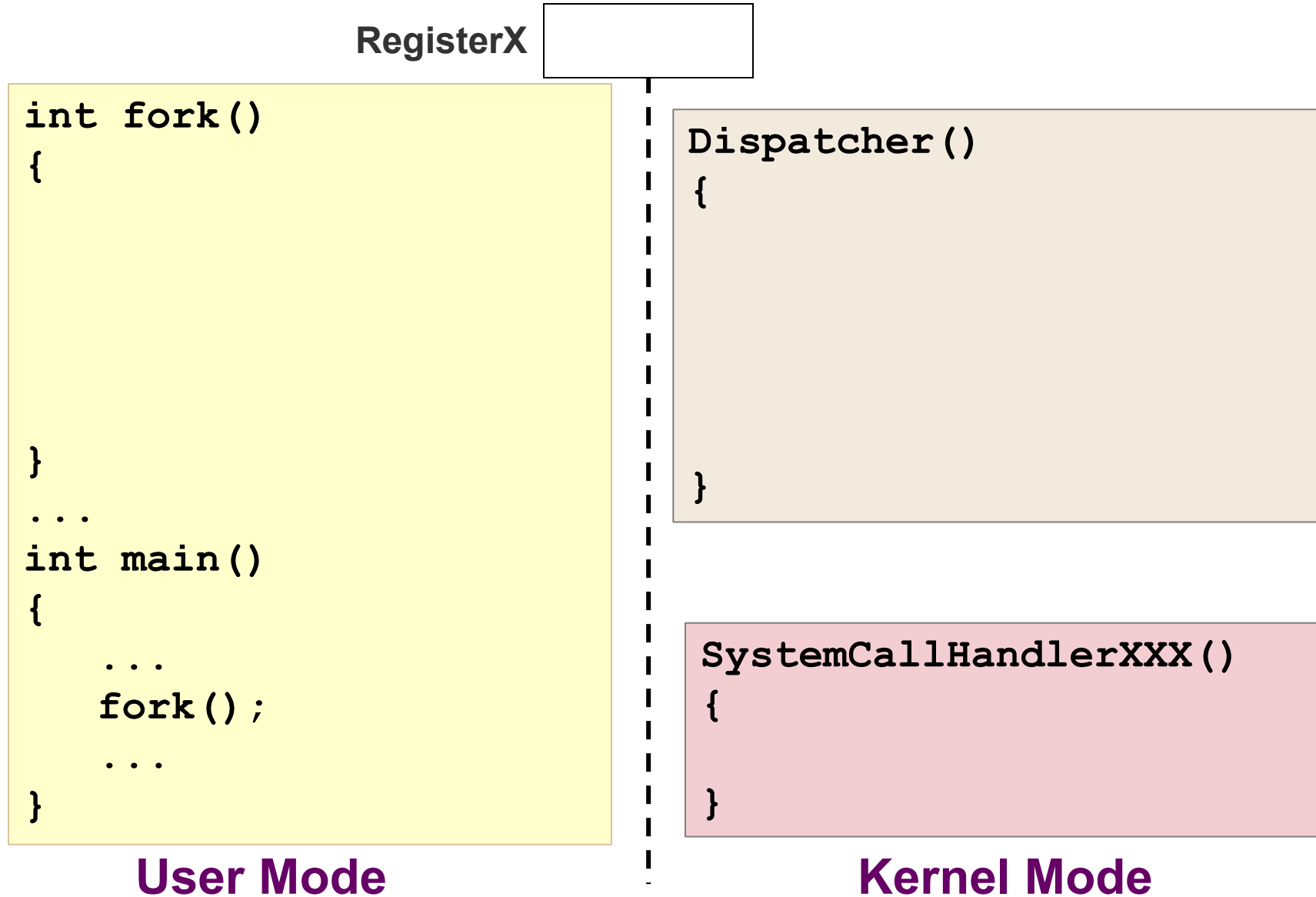
```
if I am the parent (i.e. the shell)
```

```
    wait ( child to finish )
```

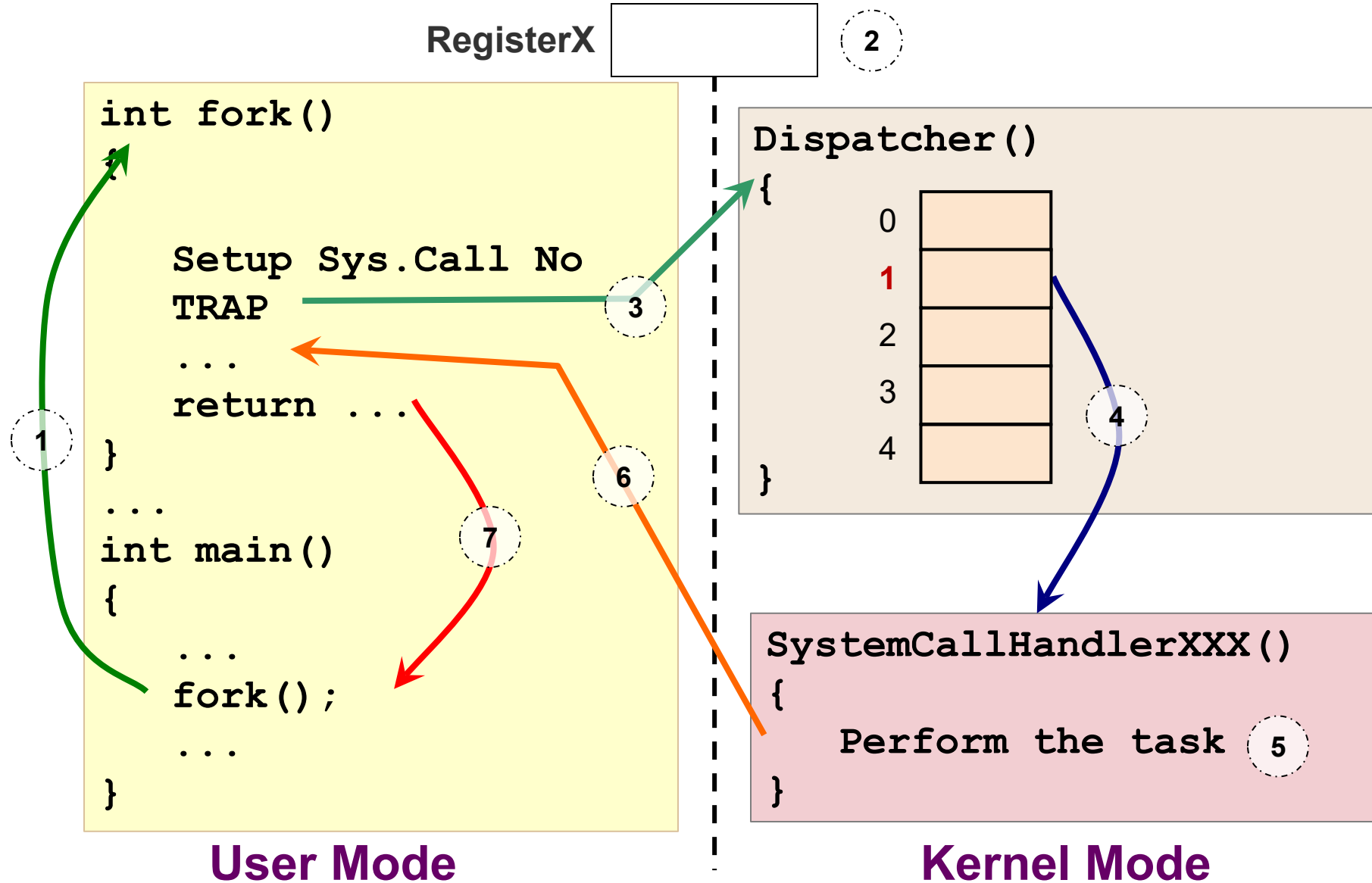
```
else
```

```
    exec ( UserCmd )
```

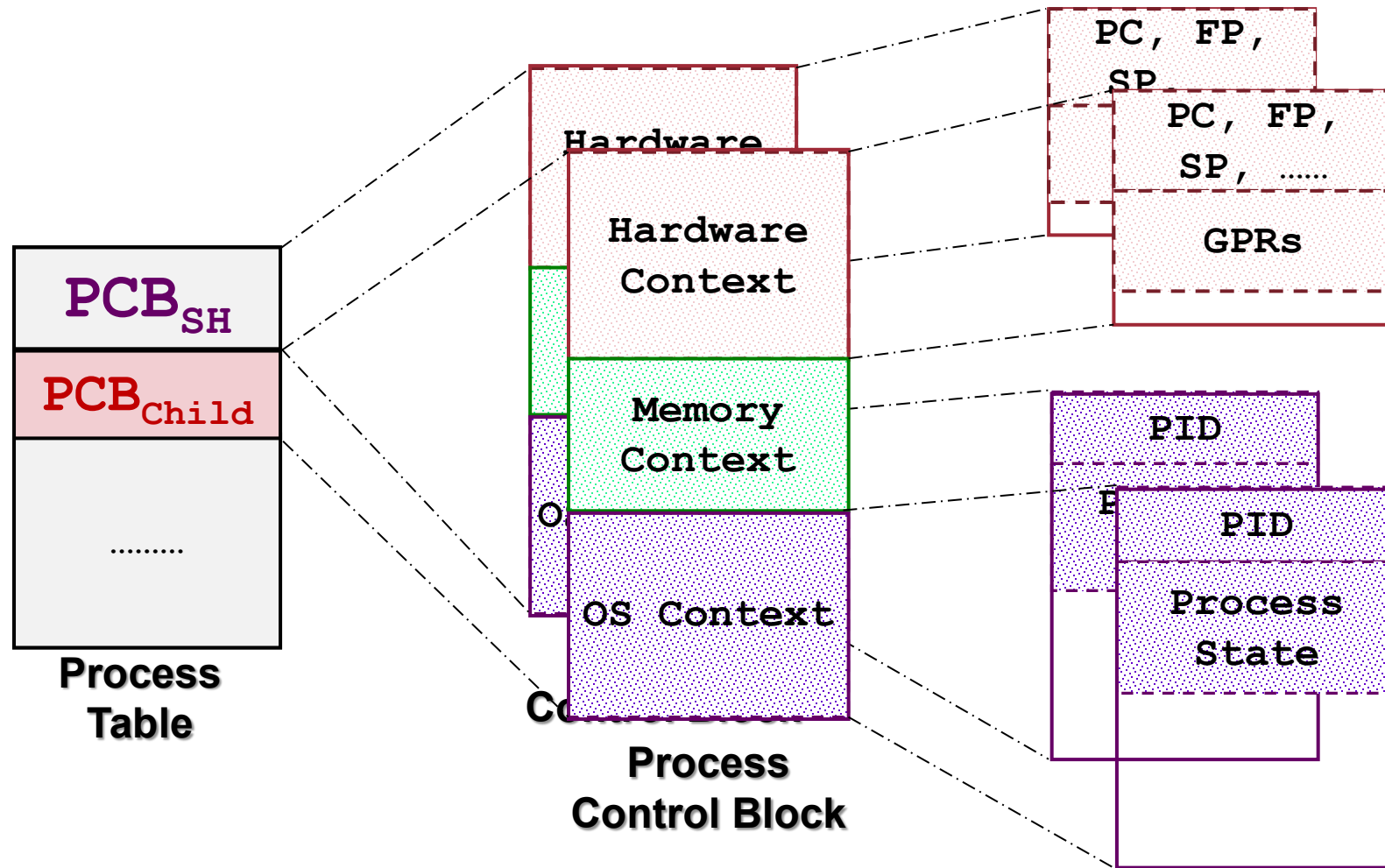
fork() involves a **system call**



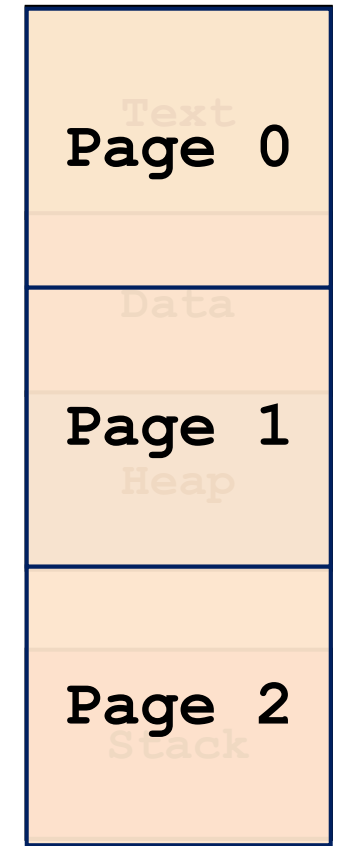
fork() involves a **system call**



What is the **effect** of `fork()`?



Memory Space of a Process



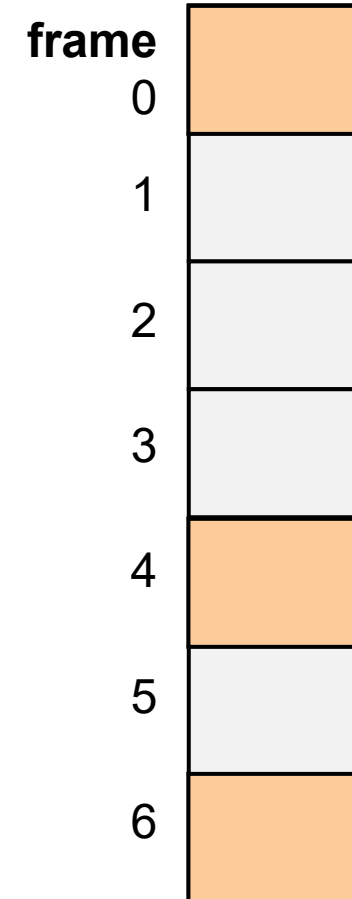
**Process
Memory Space**

0	4
1	6
2	0

Page Table

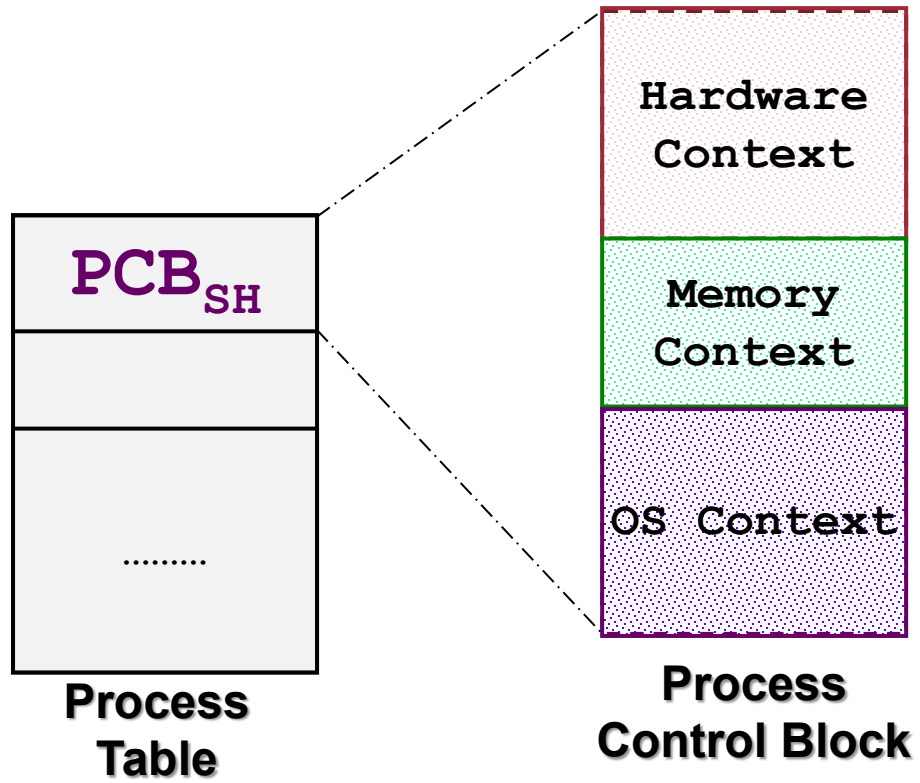
p0	f4
p2	f0

TLB

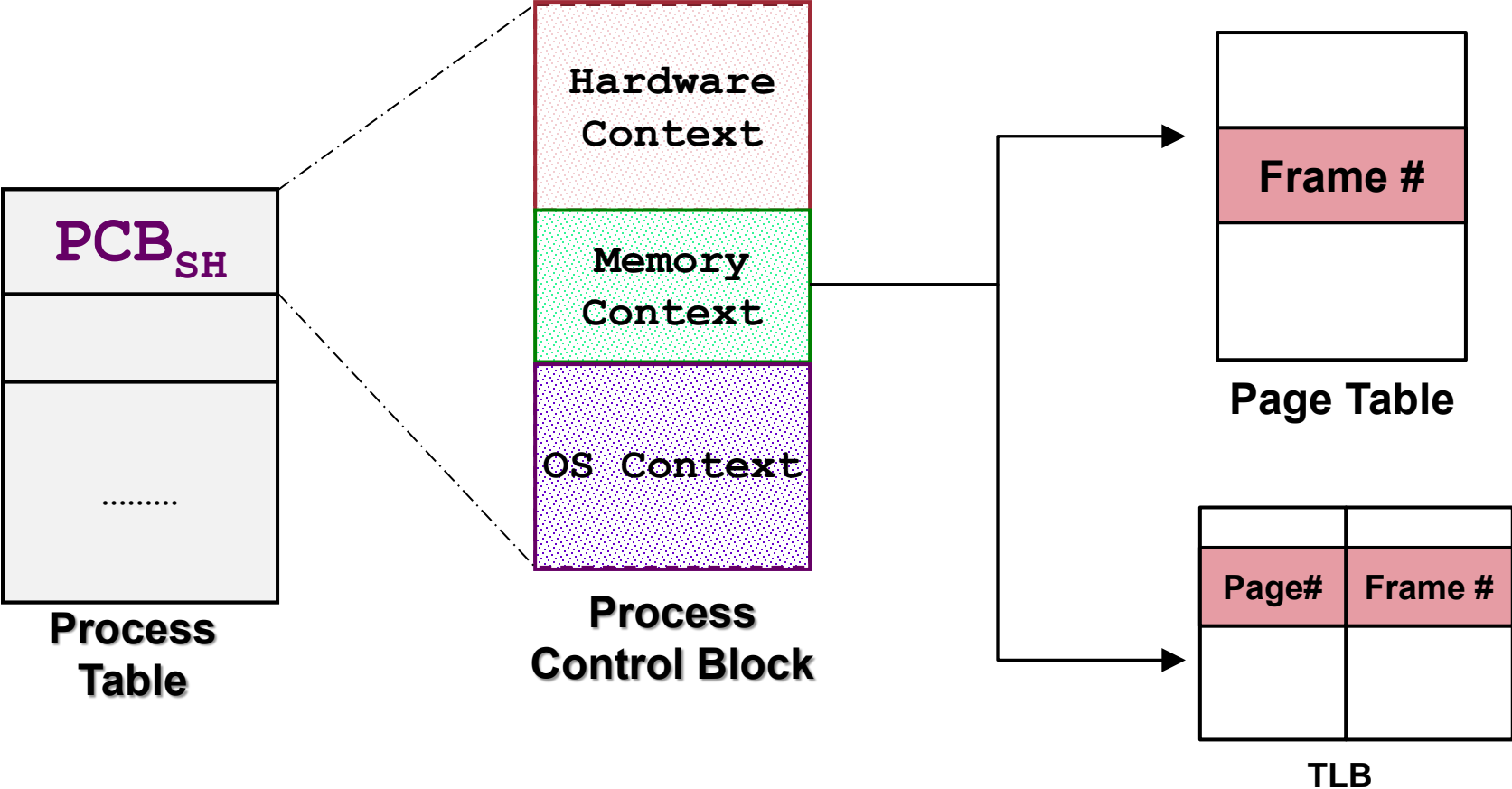


**Physical
Memory**

Memory Context = ?



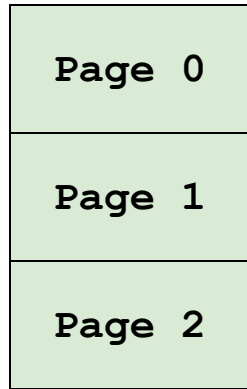
Memory Context = ?



What is the benefit of "duplication"

A

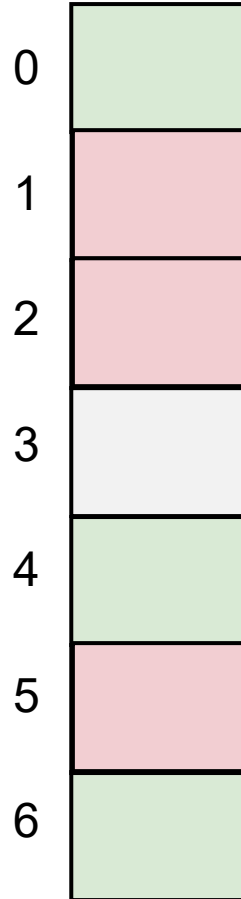
B



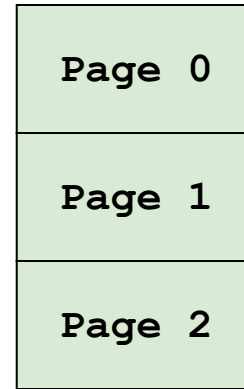
Memory Space

0	4
1	6
2	0

page table



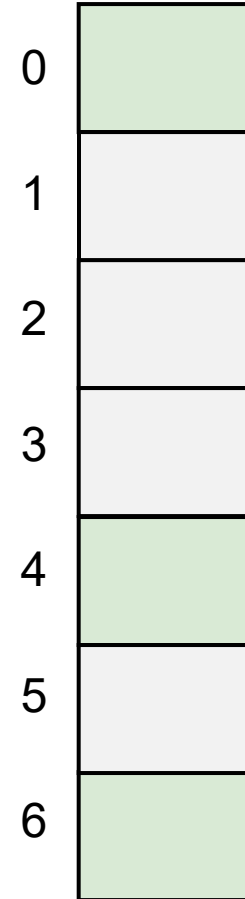
physical memory



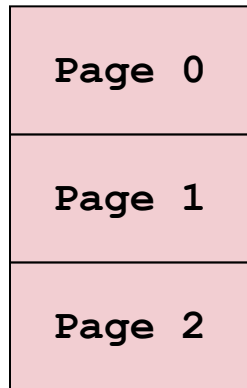
Memory Space

0	4
1	6
2	0

page table



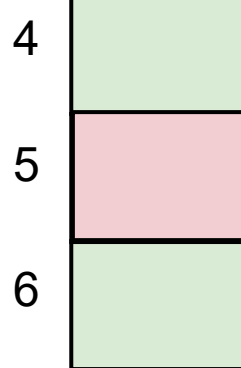
physical memory



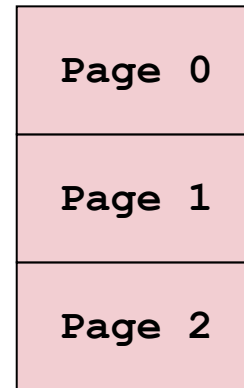
Memory Space

0	5
1	2
2	1

page table



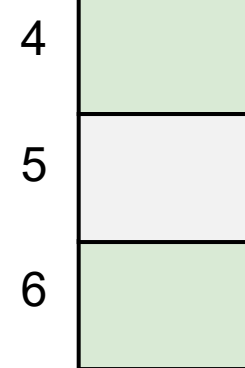
physical memory



Memory Space

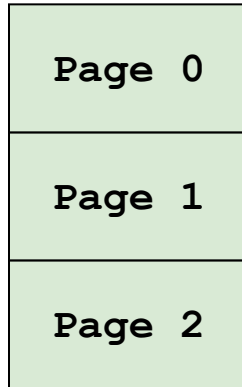
0	4
1	6
2	0

page table



physical memory

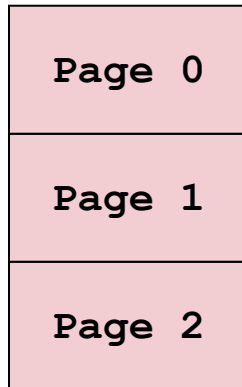
Duplicating Memory Space the **HARD WAY**



**Memory
Space**

0	4
1	6
2	0

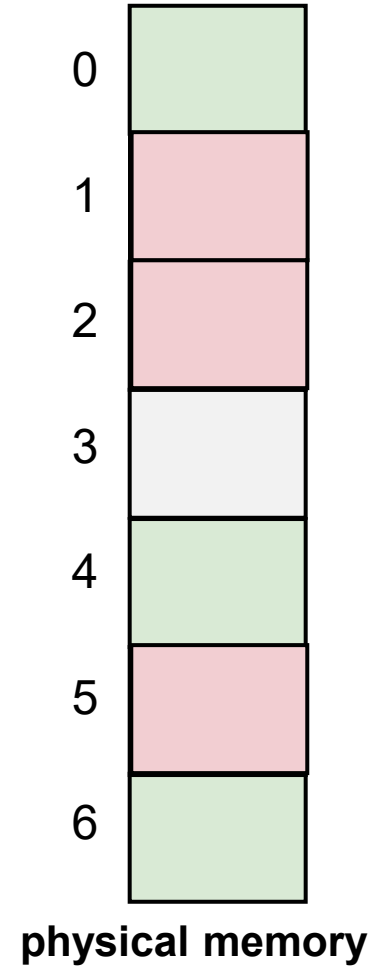
page table



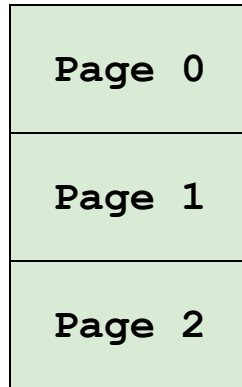
**Memory
Space**

0	5
1	2
2	1

page table



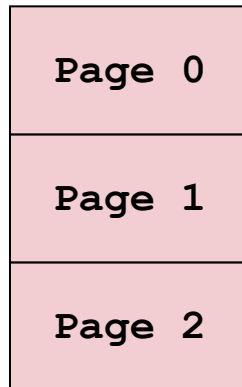
Copy on Write



Memory
Space

0	4
1	6
2	0

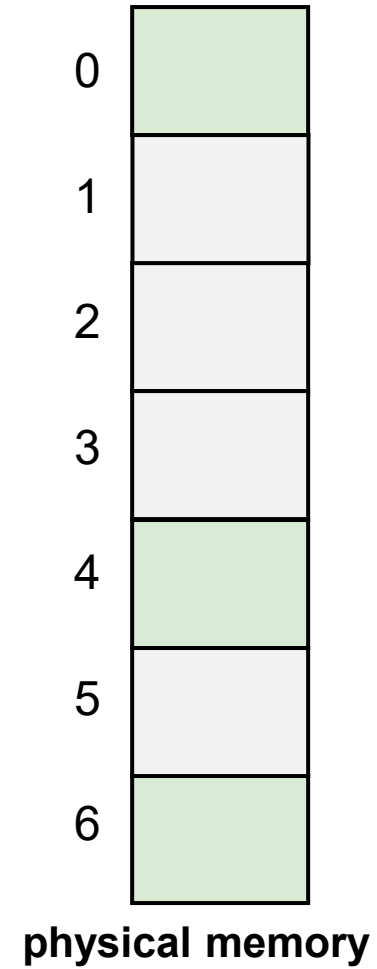
page table



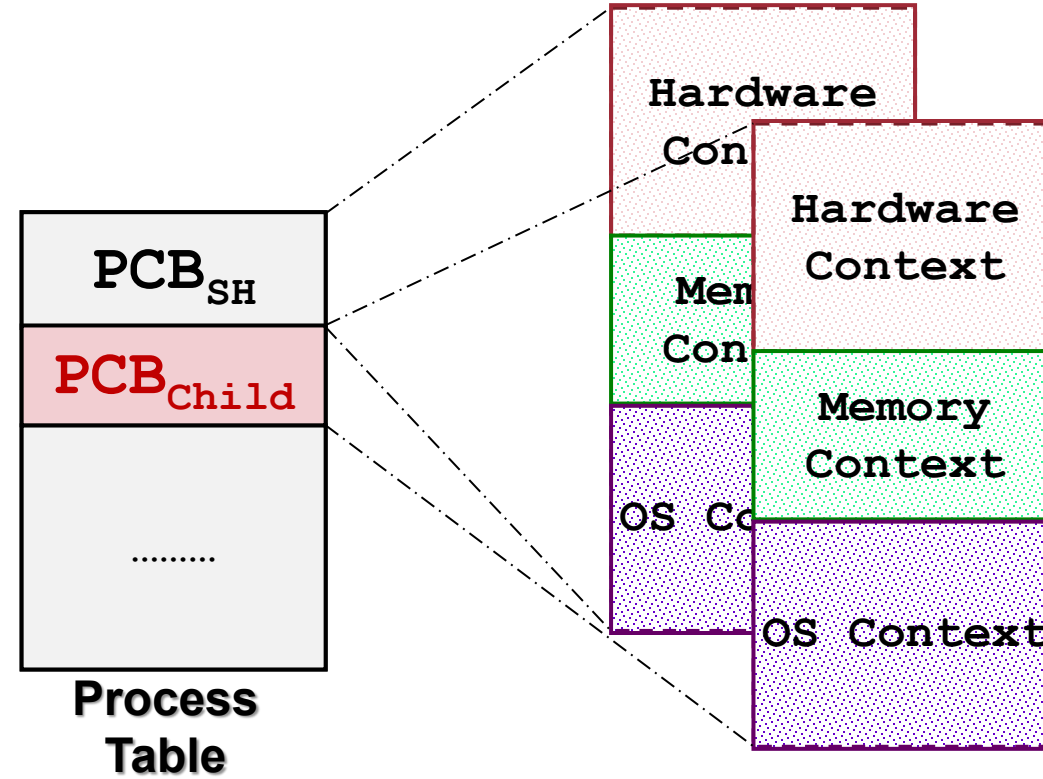
Memory
Space

0	4
1	6
2	0

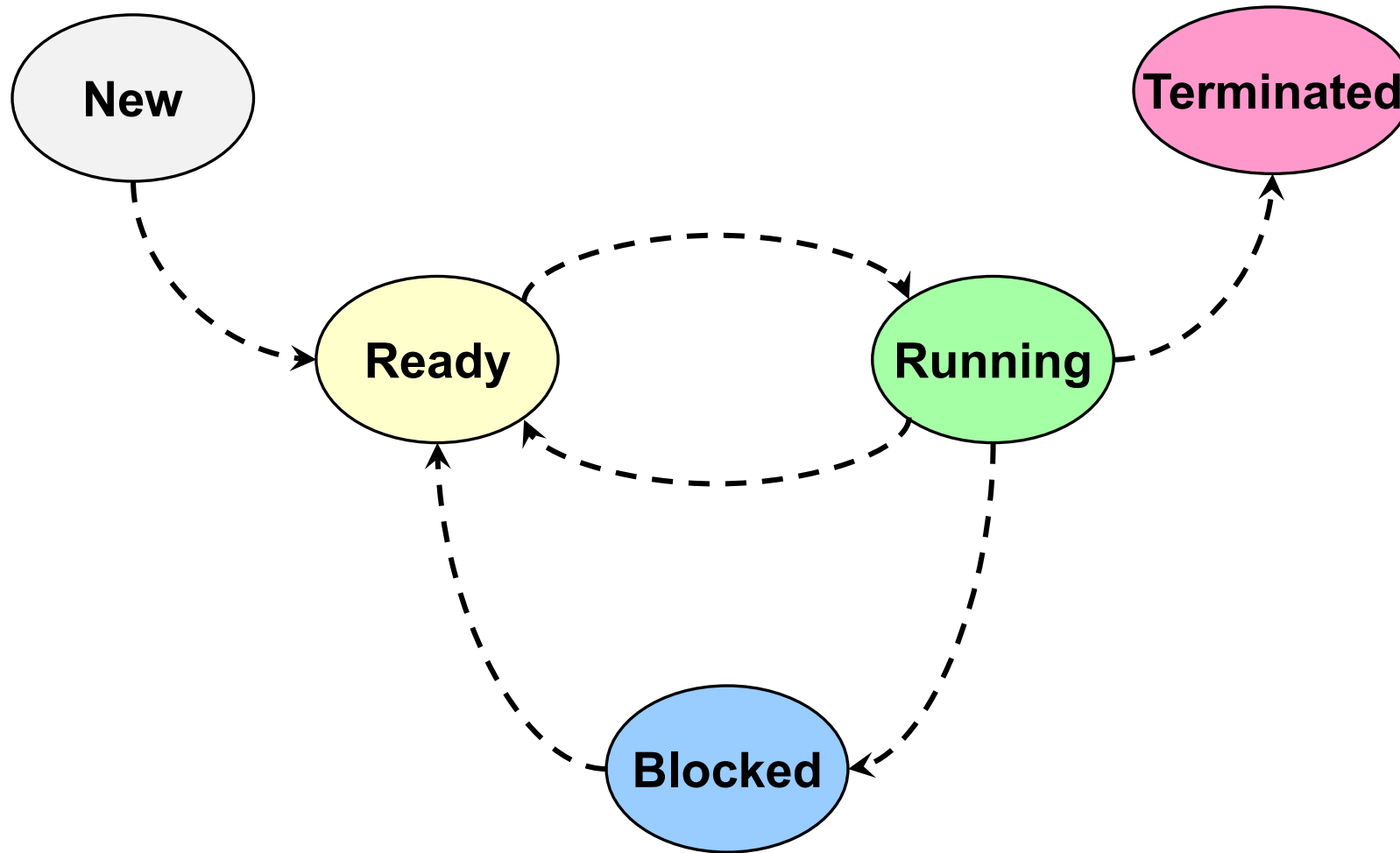
page table



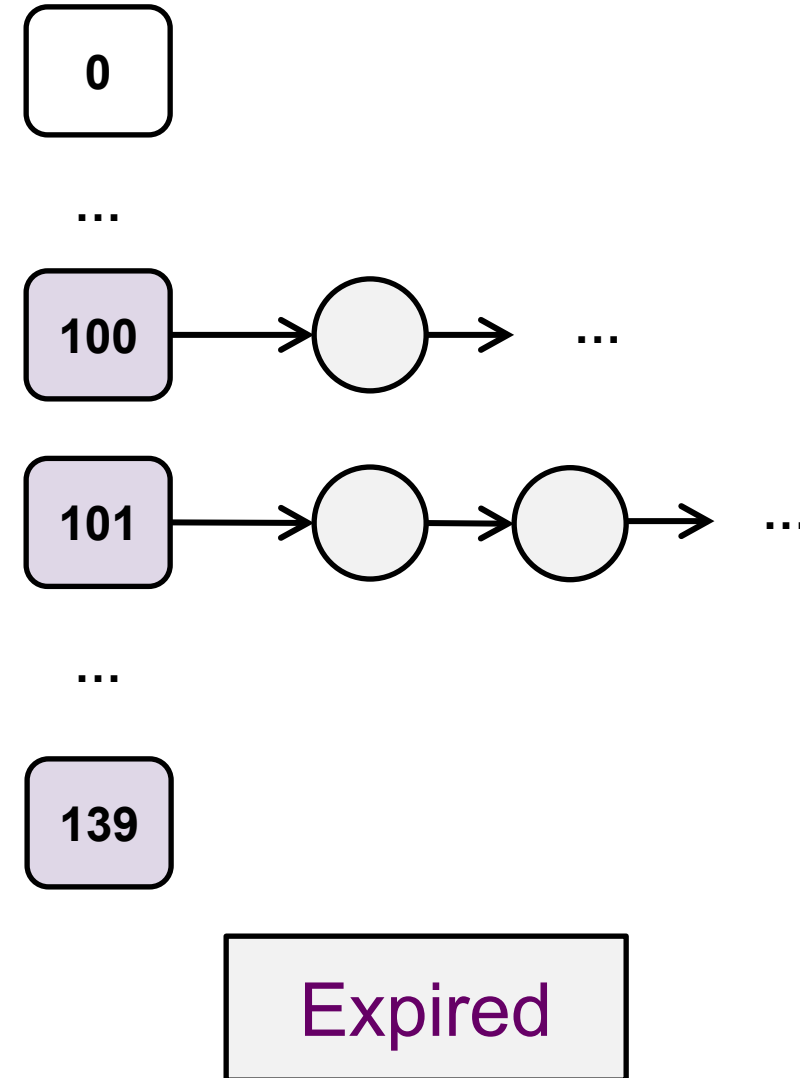
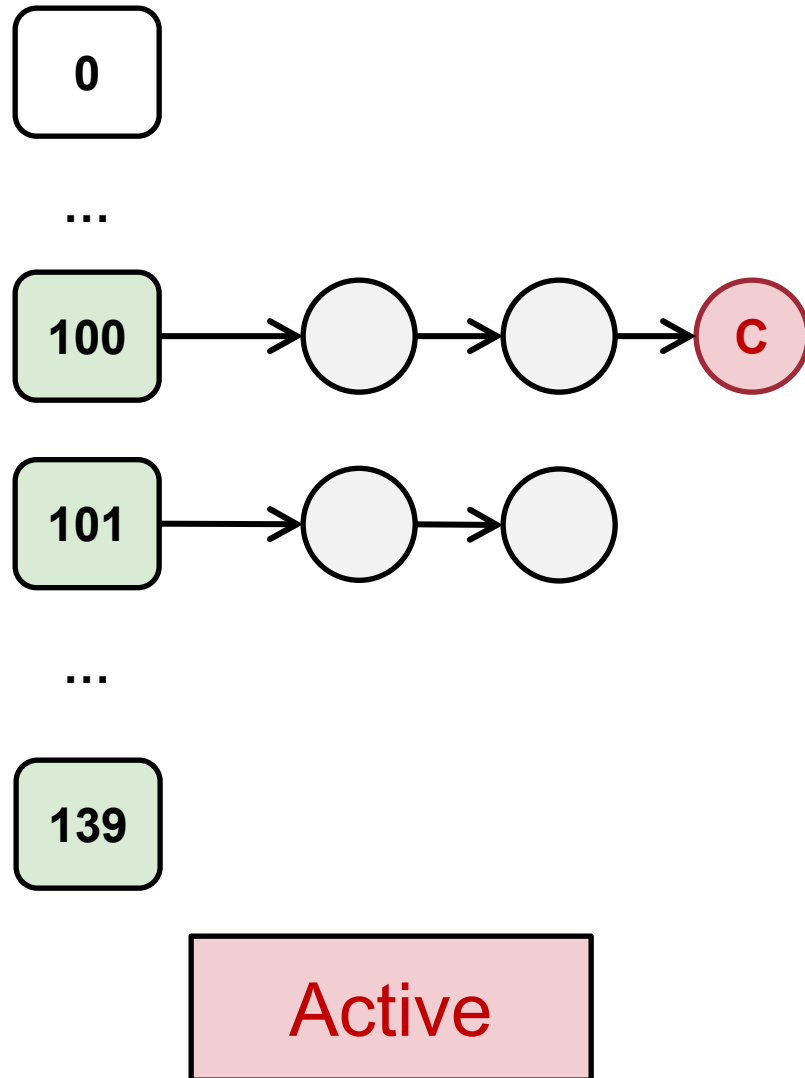
So, Effect of **fork**()



Child, welcome to the world!



Child, welcome to the Queue!



fork() finishes and returns

RegisterX

```
int fork()
{
    Setup Sys.Call No
    TRAP
    ...
    return ...
}
...
int main()
{
    ...
    fork();
    ...
}
```

User Mode

Dispatcher()

```
{
    0
    1
    2
    3
    4
}
```

SystemCallHandlerXXX()

```
{
    Perform the task
}
```

Kernel Mode

Typical steps for **Shell Interpreter**

```
UserCmd ← Read from keyboard
```

```
fork()
```

```
if I am the parent (i.e. the shell)
```

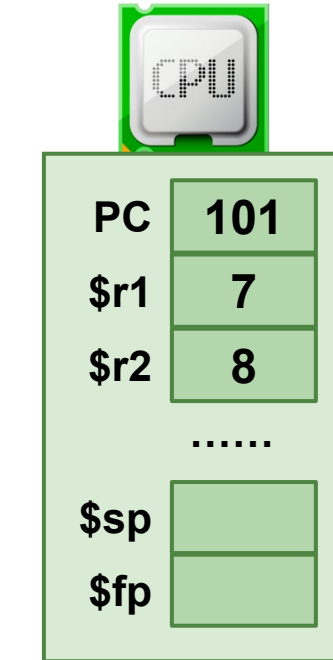
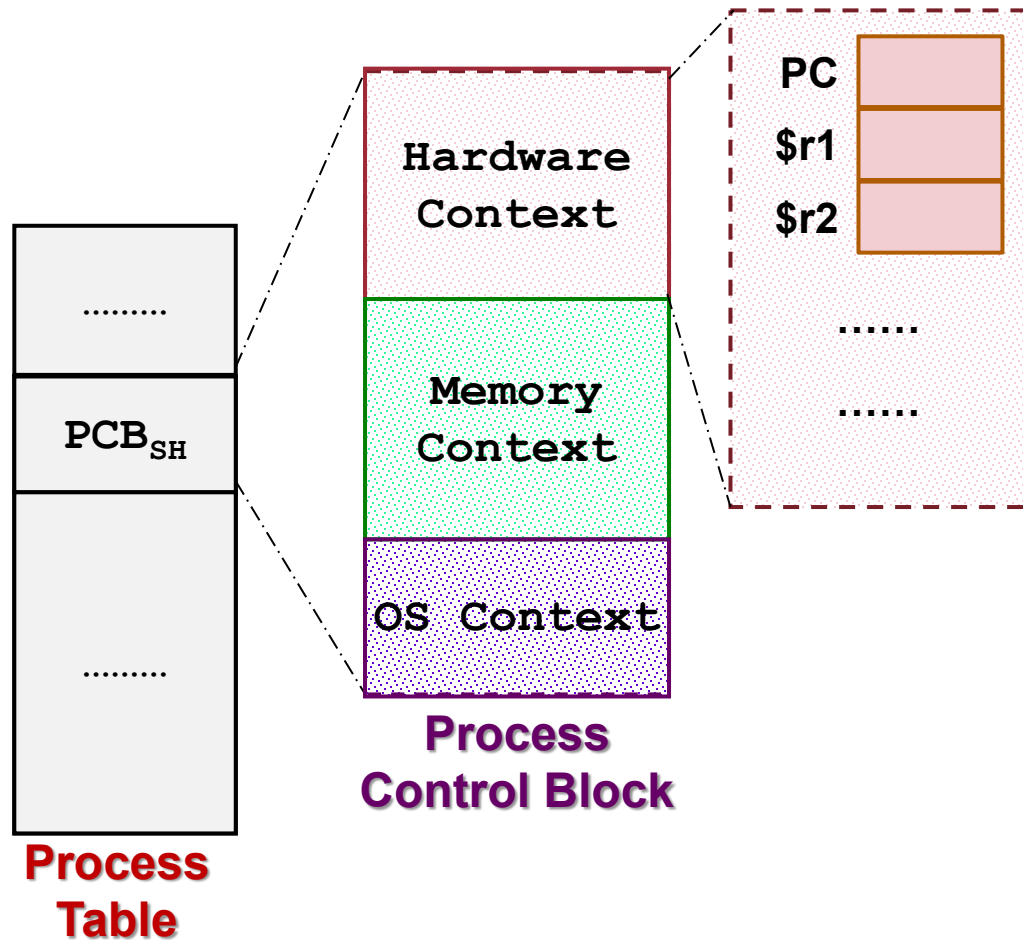
```
    wait ( child to finish )
```

```
else
```

```
    exec ( UserCmd )
```



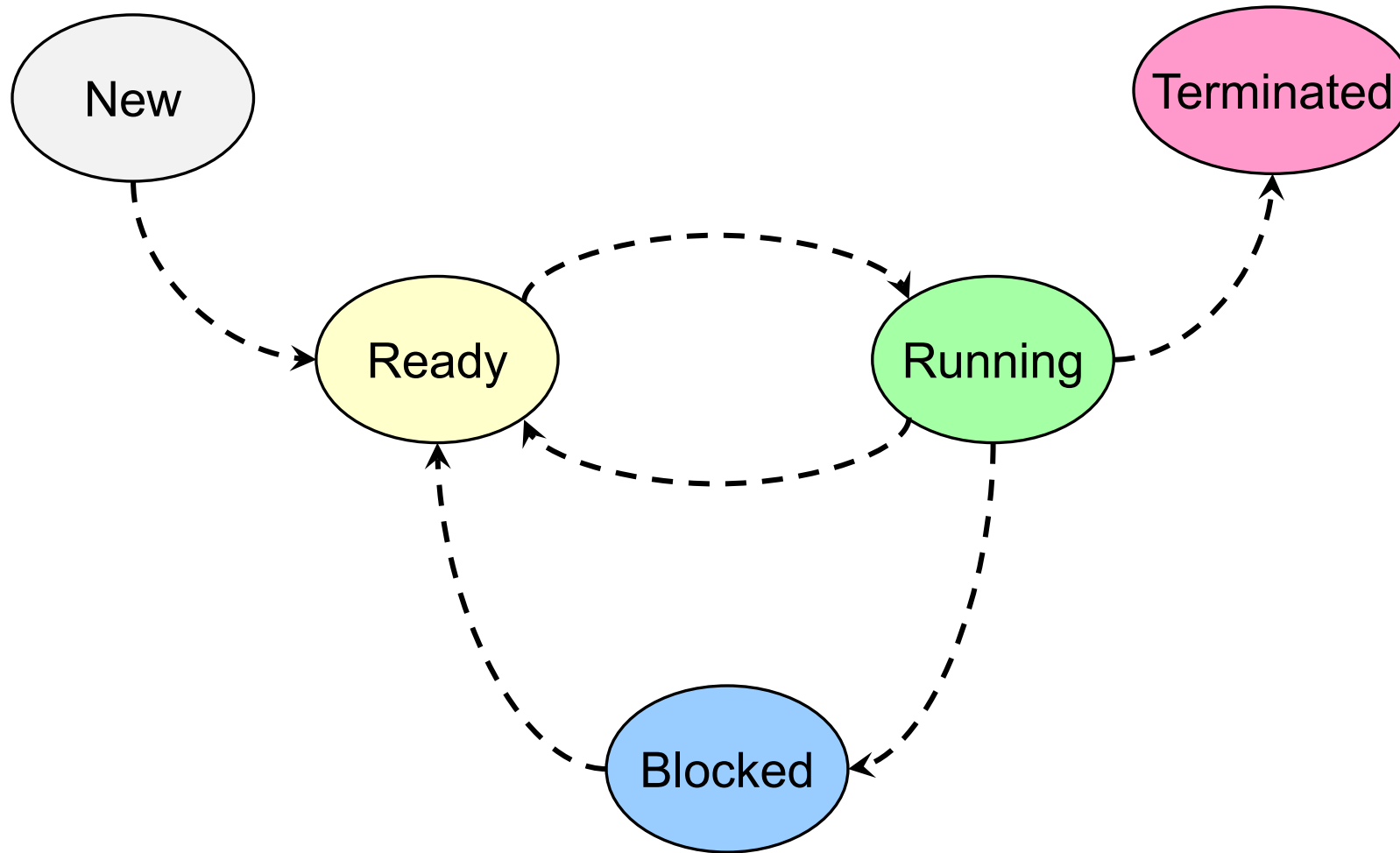
wait(): The interpreter will....



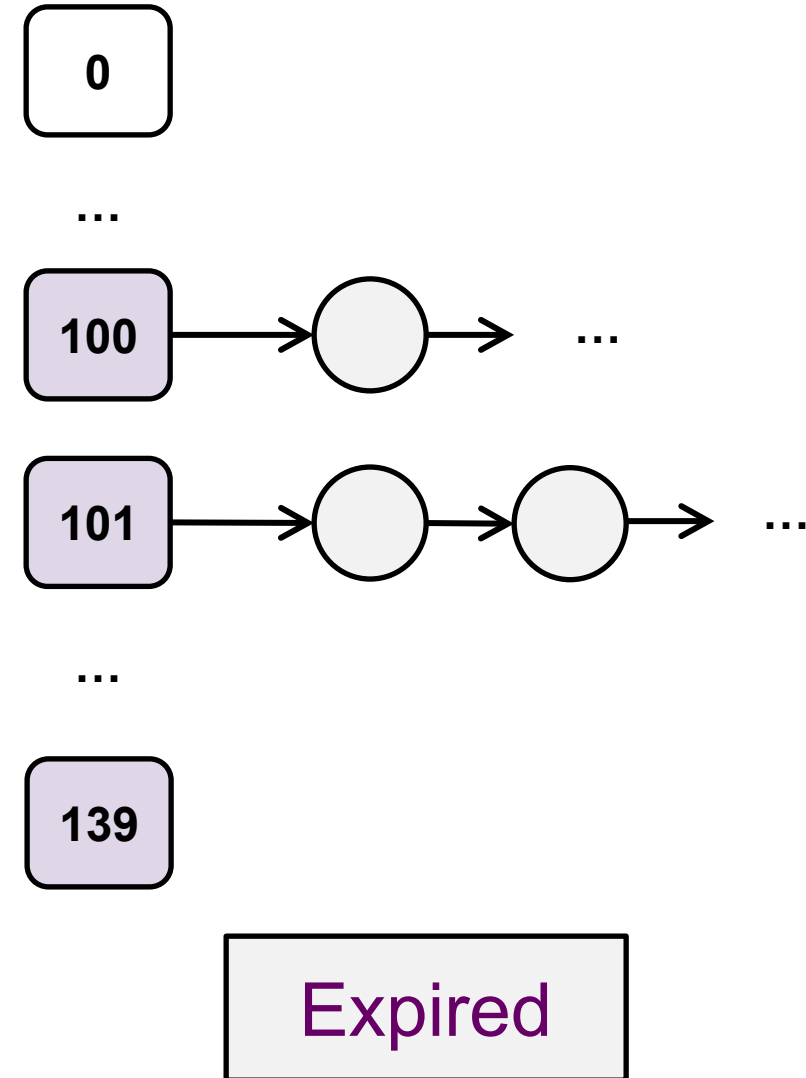
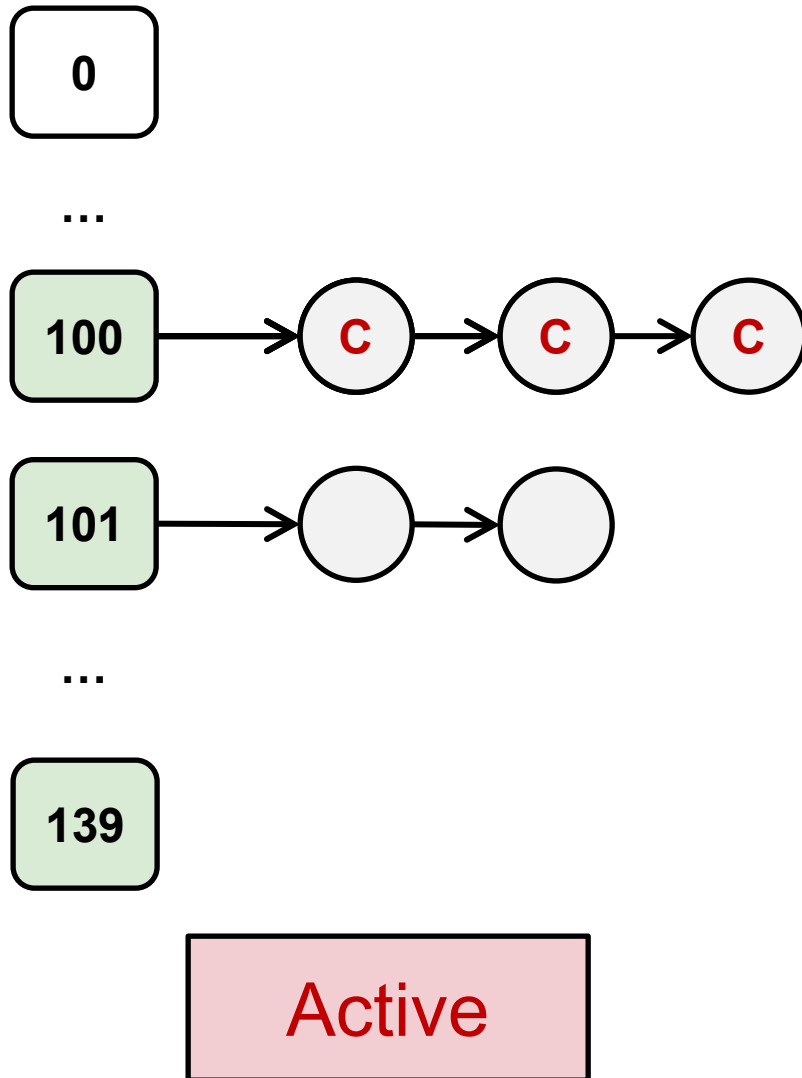
p0	f4
p2	f0

TLB

The interpreter is now...



Hmm... **who** gets to **run**?



Typical steps for **Shell Interpreter**

UserCmd ← Read from keyboard

fork()

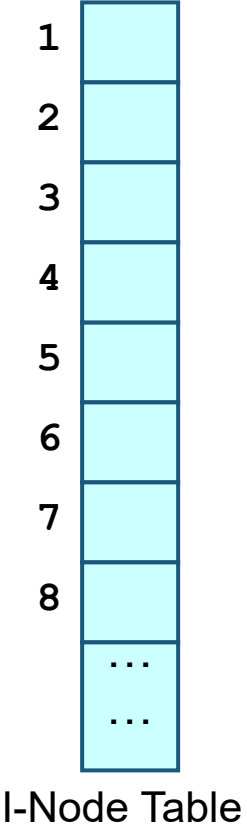
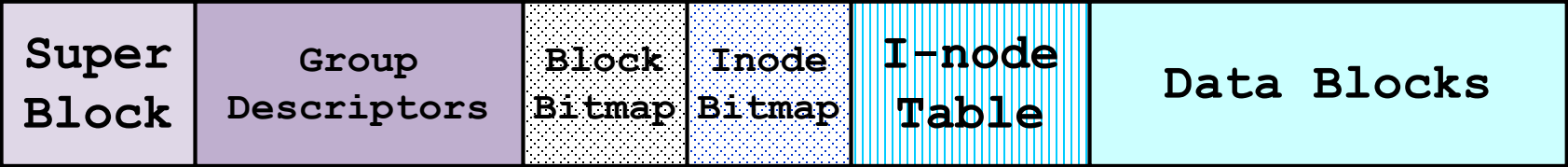
if I am the parent (i.e. the shell)

wait (child to finish)

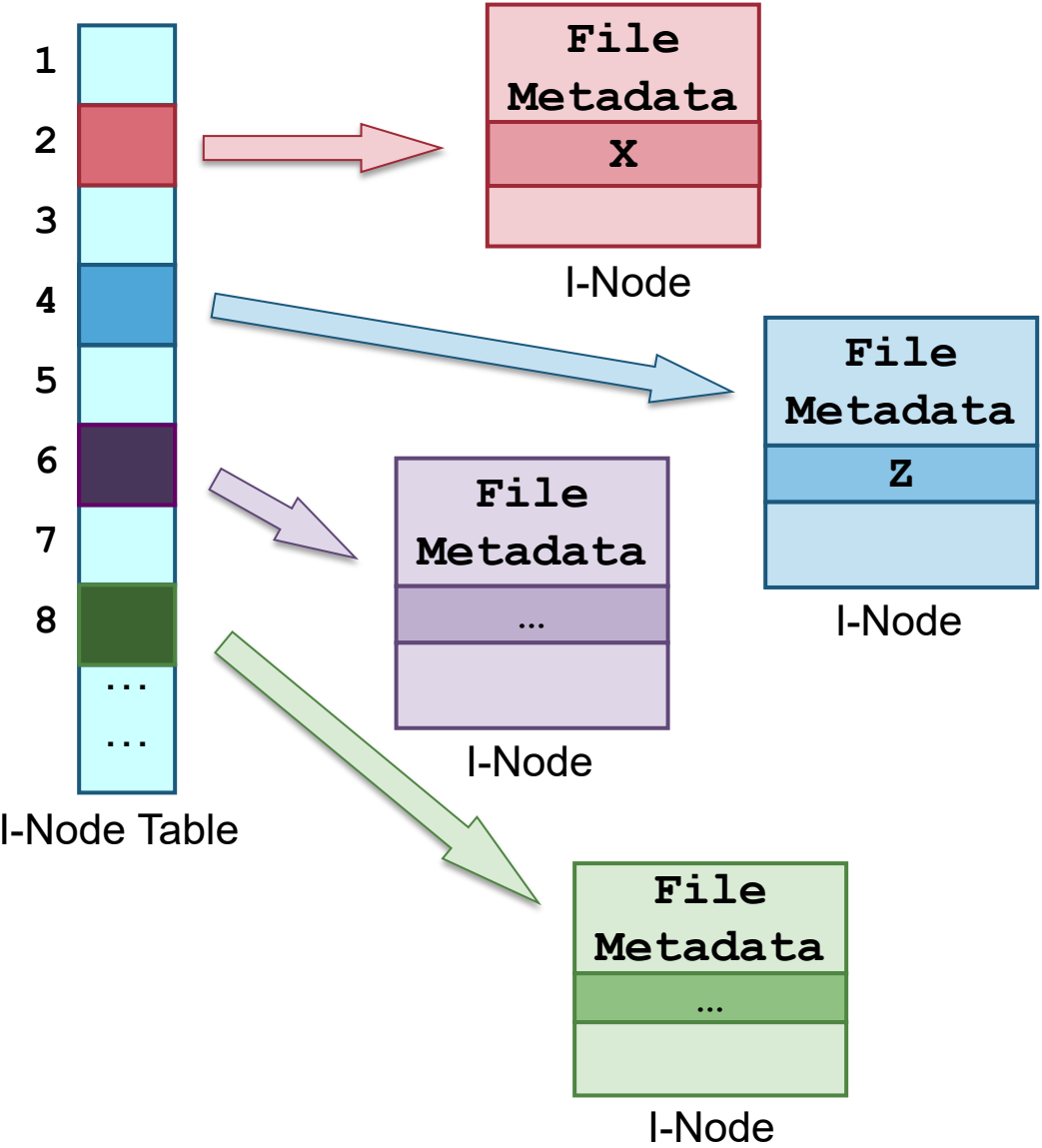
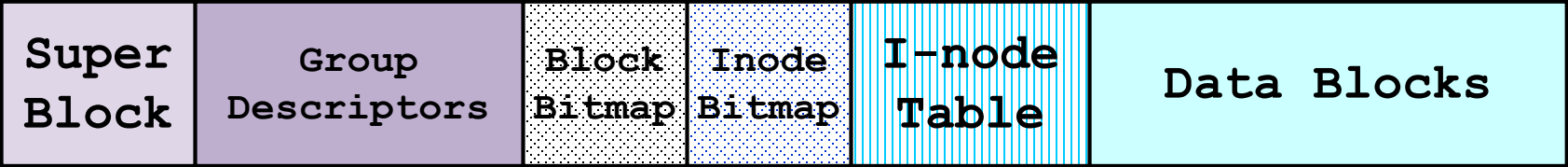
else

exec (UserCmd)

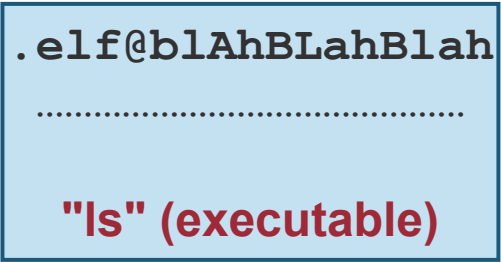
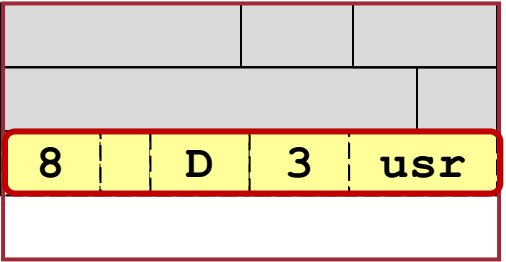




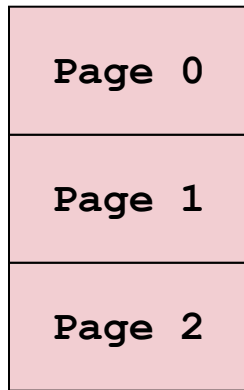
/usr/bin/ls



/usr/bin/ls



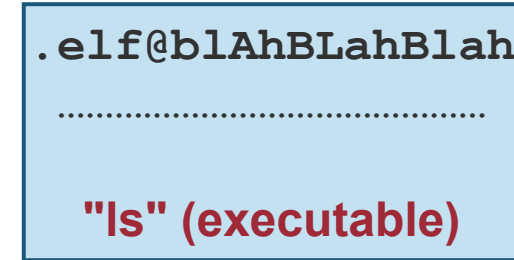
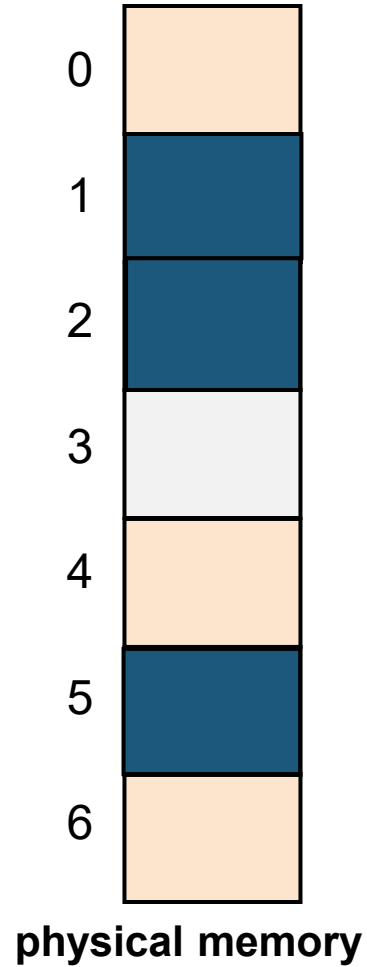
Memory Content Replaced



**Memory
Space**

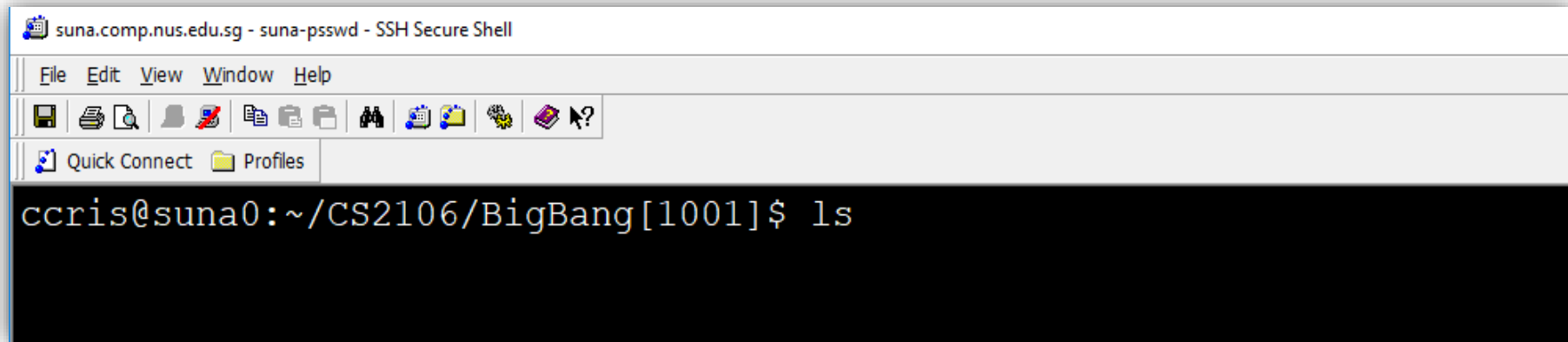
0	5
1	2
2	1

page table



Disk Block Z

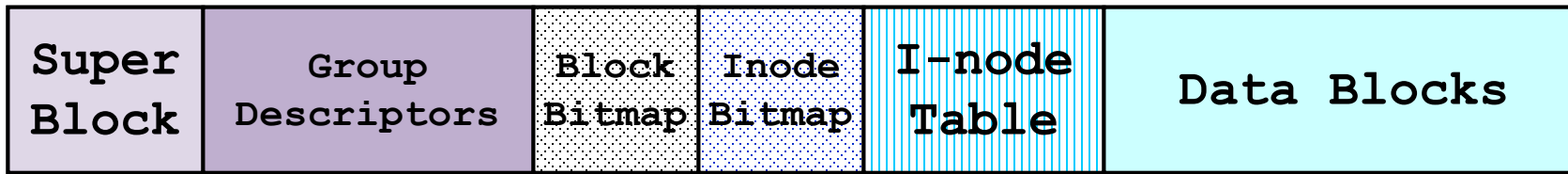
Child is now "**ls**", what next?



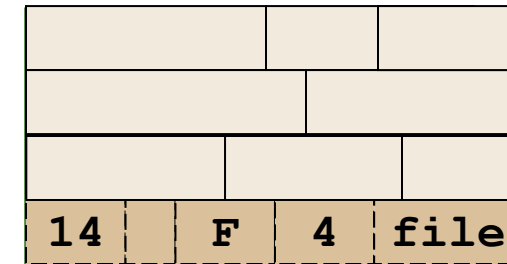
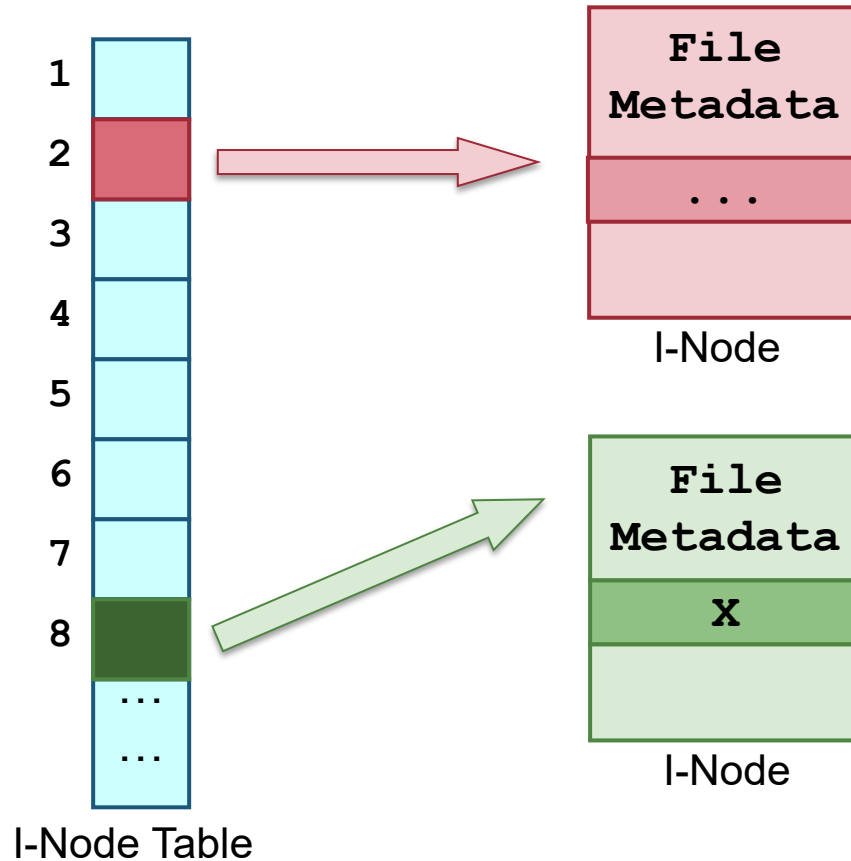
The screenshot shows a terminal window titled 'suna.comp.nus.edu.sg - suna-psswd - SSH Secure Shell'. The window has a menu bar with 'File', 'Edit', 'View', 'Window', and 'Help'. Below the menu bar is a toolbar with various icons. The terminal area is black with white text. The prompt is 'ccris@suna0:~/CS2106/BigBang[1001]\$' and the command 'ls' has been entered.

```
suna.comp.nus.edu.sg - suna-psswd - SSH Secure Shell
File Edit View Window Help
[Toolbar icons]
Quick Connect Profiles
ccris@suna0:~/CS2106/BigBang[1001]$ ls
```

Listing **/.../.../BigBang**

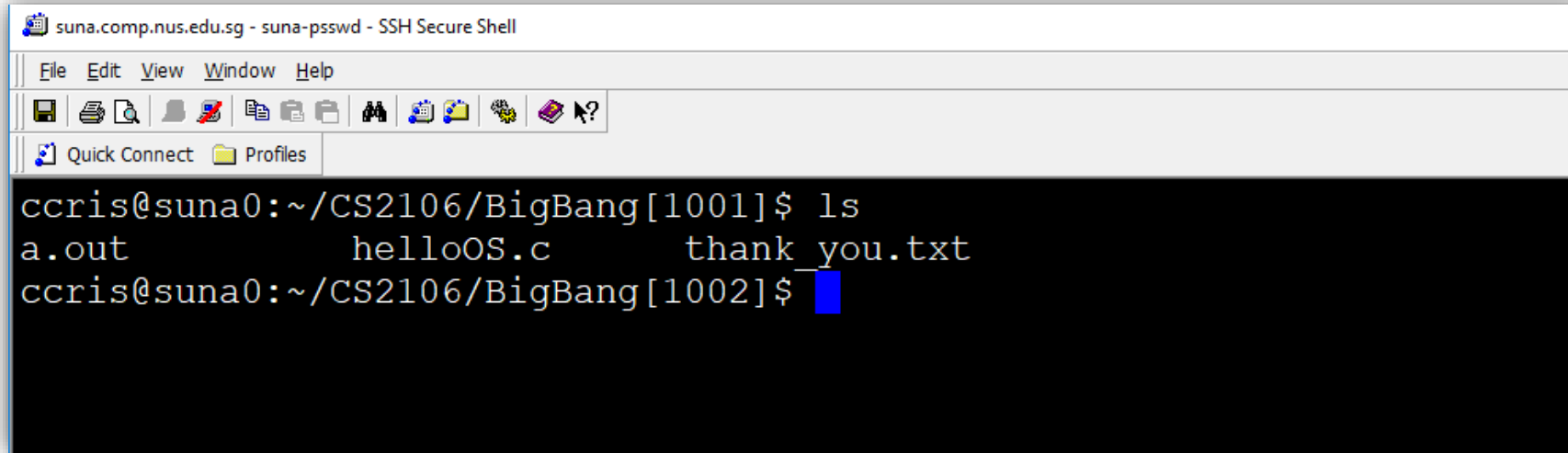


Listing **/.../.../BigBang**



ls -l

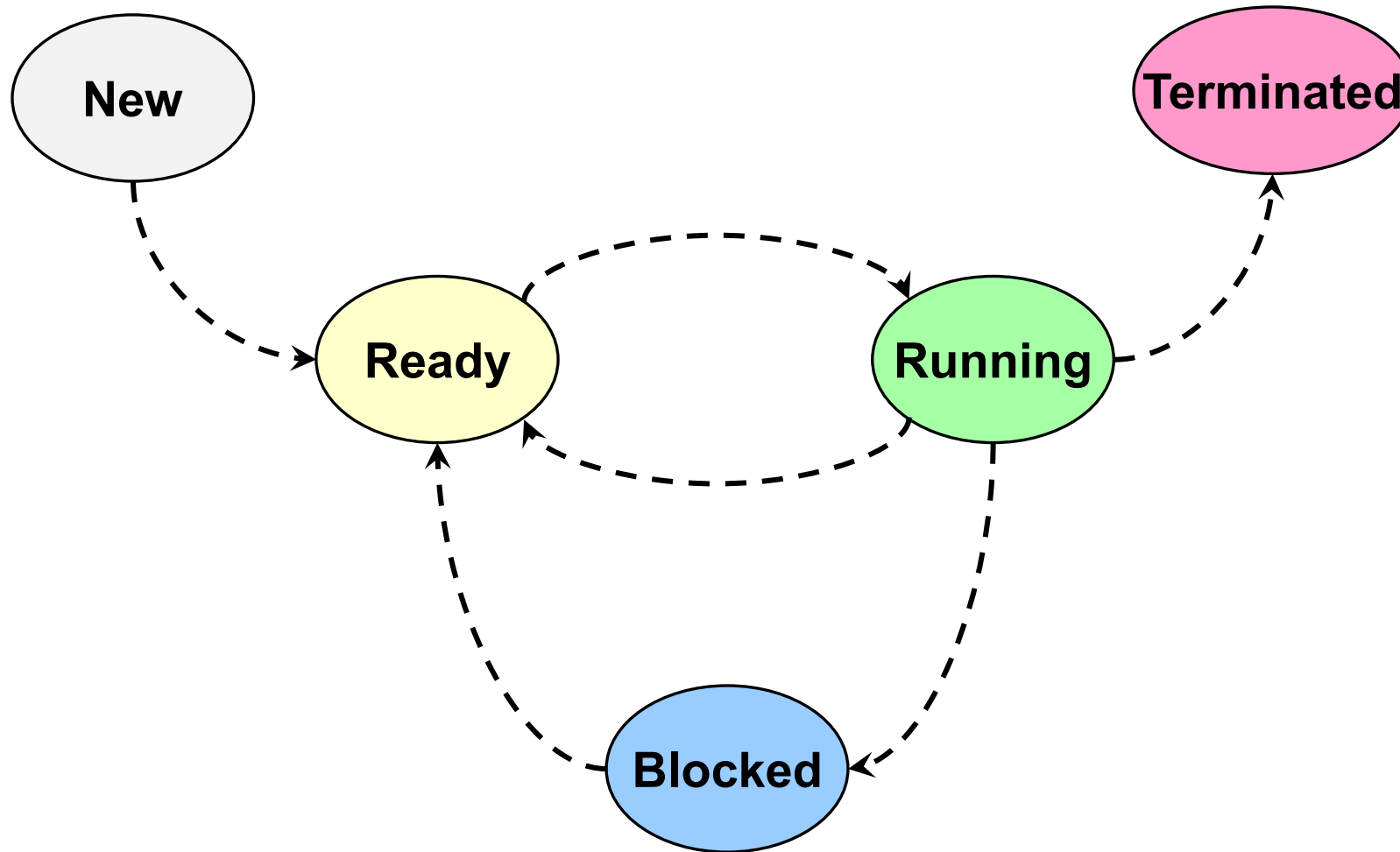
"ls" prints the directory content



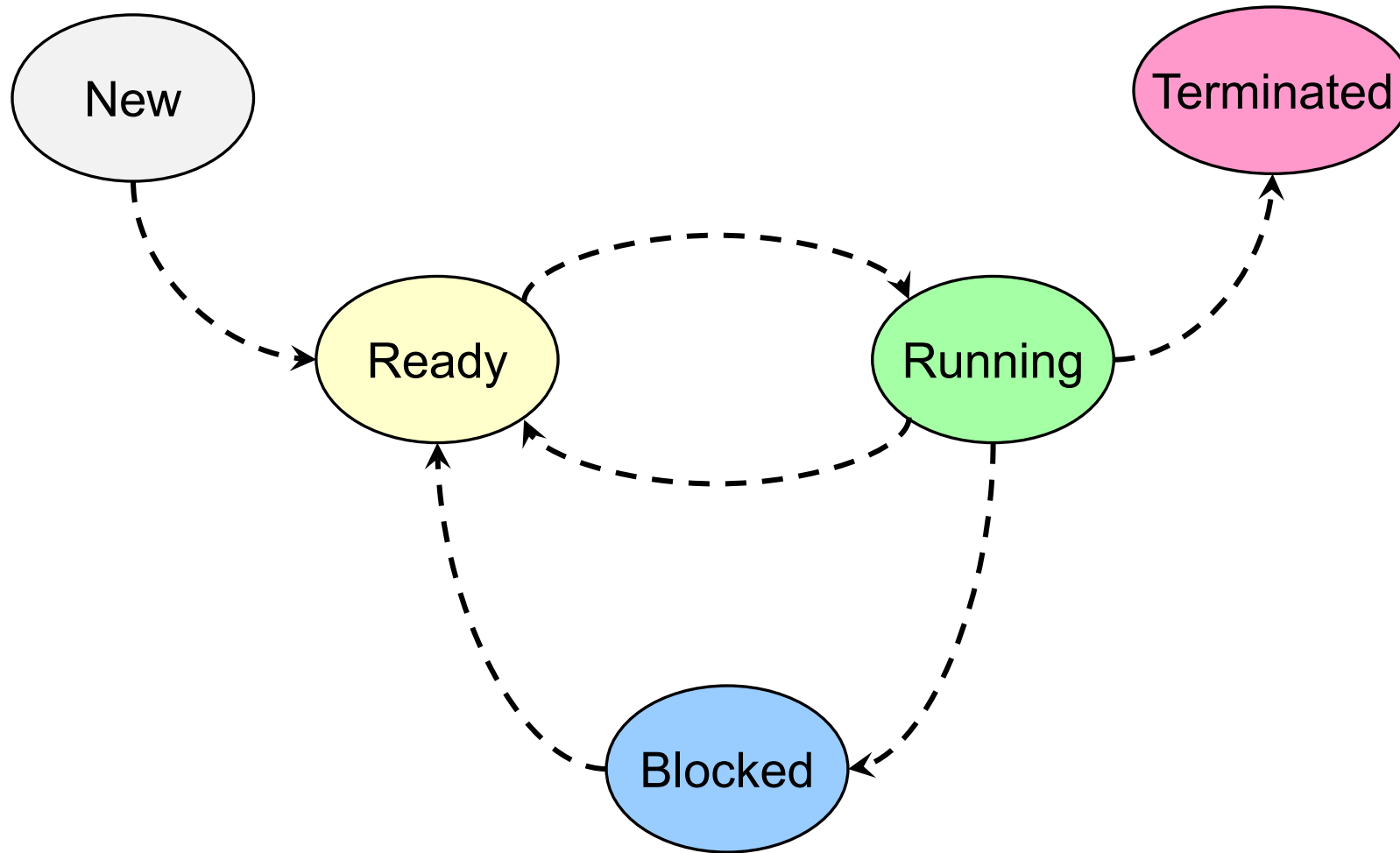
The screenshot shows a terminal window titled "suna.comp.nus.edu.sg - suna-psswd - SSH Secure Shell". The window has a menu bar with "File", "Edit", "View", "Window", and "Help". Below the menu bar is a toolbar with various icons. The terminal content shows the user "ccris" at host "suna0" in the directory "~/CS2106/BigBang". The user enters the command "ls" and the output is displayed on the next line: "a.out", "helloOS.c", and "thank_you.txt". The prompt changes to "[1002]" after the command is executed.

```
suna.comp.nus.edu.sg - suna-psswd - SSH Secure Shell
File Edit View Window Help
a.out helloOS.c thank_you.txt
ccris@suna0:~/CS2106/BigBang[1001]$ ls
a.out helloOS.c thank_you.txt
ccris@suna0:~/CS2106/BigBang[1002]$
```

Child **exits**



The interpreter **hears about it**....



Typical steps for **Shell Interpreter**

UserCmd ← Read from keyboard



fork()

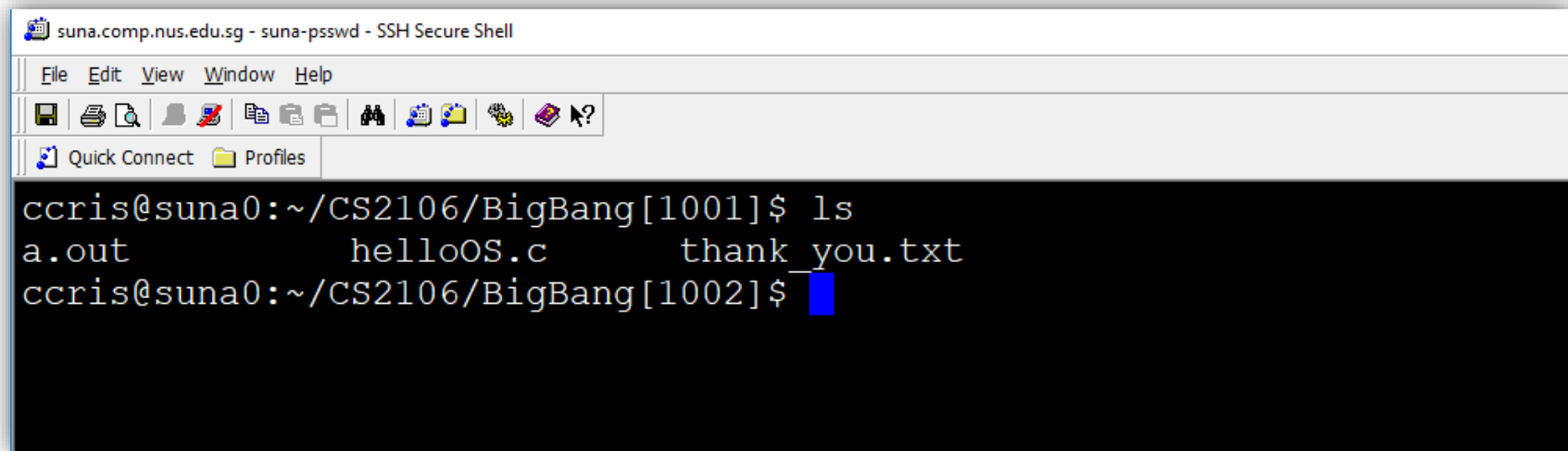
if I am the parent (i.e. the shell)

 wait (child to finish)

else

exec(UserCmd)

Woohoo!!

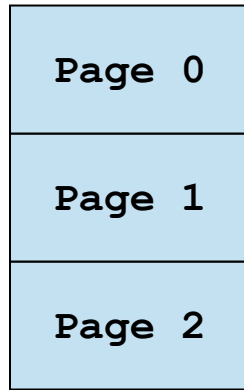


The image shows a screenshot of an SSH terminal window. The title bar reads "suna.comp.nus.edu.sg - suna-psswd - SSH Secure Shell". The menu bar includes "File", "Edit", "View", "Window", and "Help". The toolbar contains icons for file operations like save, print, and copy. Below the toolbar, there are tabs for "Quick Connect" and "Profiles". The terminal area has a black background with white text. The prompt is "ccris@suna0:~/CS2106/BigBang[1001]". The user has entered the command "ls", and the output is "a.out", "helloOS.c", and "thank_you.txt". The prompt has changed to "[1002]" and a blue cursor is visible.

```
suna.comp.nus.edu.sg - suna-psswd - SSH Secure Shell
File Edit View Window Help
[Icons]
Quick Connect Profiles
ccris@suna0:~/CS2106/BigBang[1001]$ ls
a.out          helloOS.c      thank_you.txt
ccris@suna0:~/CS2106/BigBang[1002]$
```

WE SHOULD SHARE!

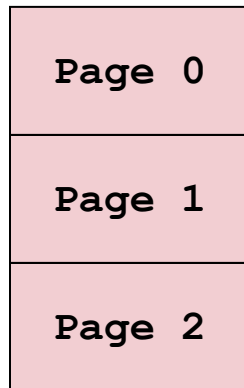
How to **Share Memory**?



**Memory
Space**

0	4
1	6
2	0

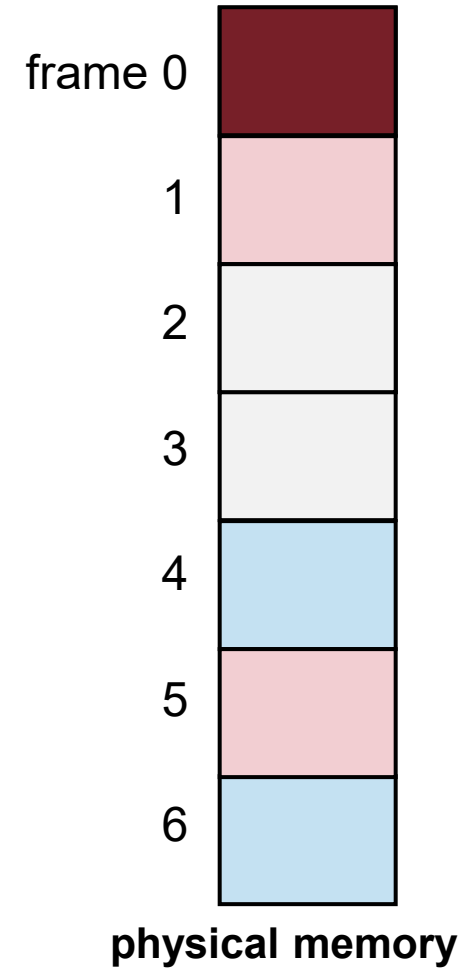
page table



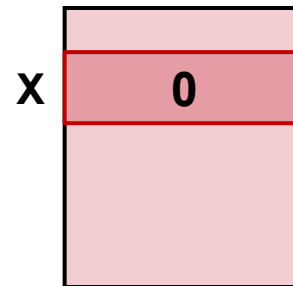
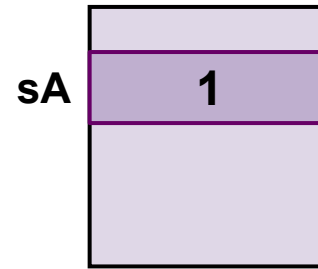
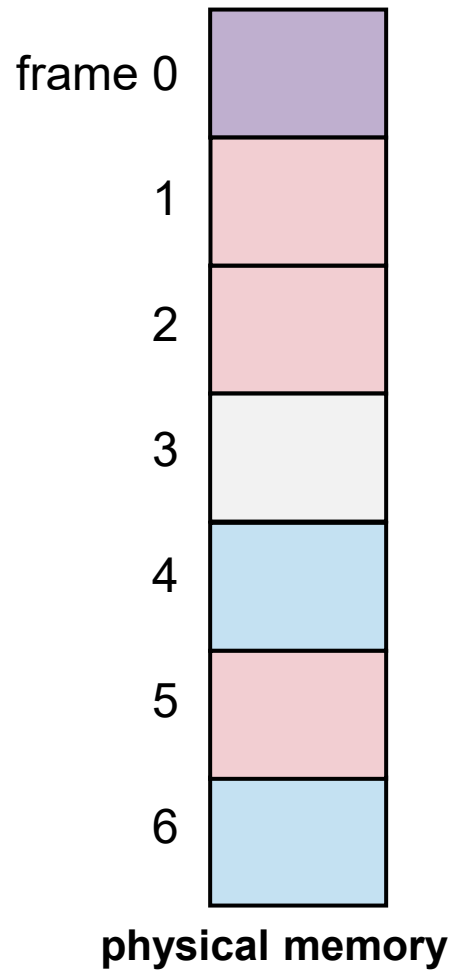
**Memory
Space**

0	5
1	0
2	1

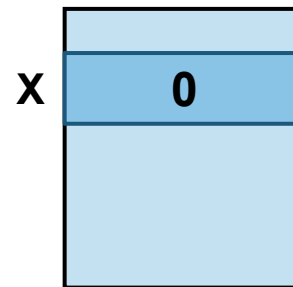
page table



($X = X + 1;$ $sA = sA + 1;$)

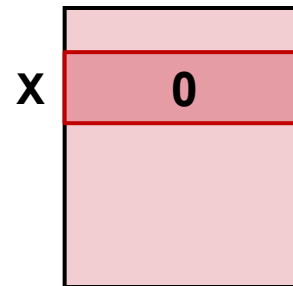
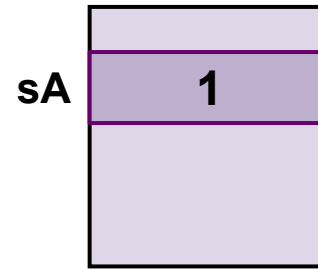
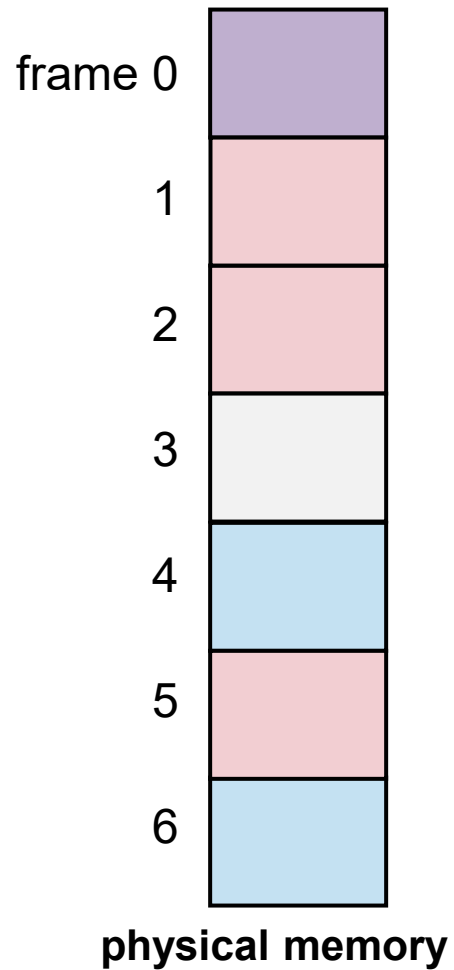


$X = X + 1;$
 $sA = sA + 1;$

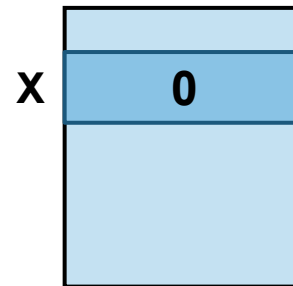


$X = X + 1;$
 $sA = sA + 1;$

Semaphore to the rescue!



```
X = X + 1;  
sA = sA + 1;
```



```
X = X + 1;  
sA = sA + 1;
```

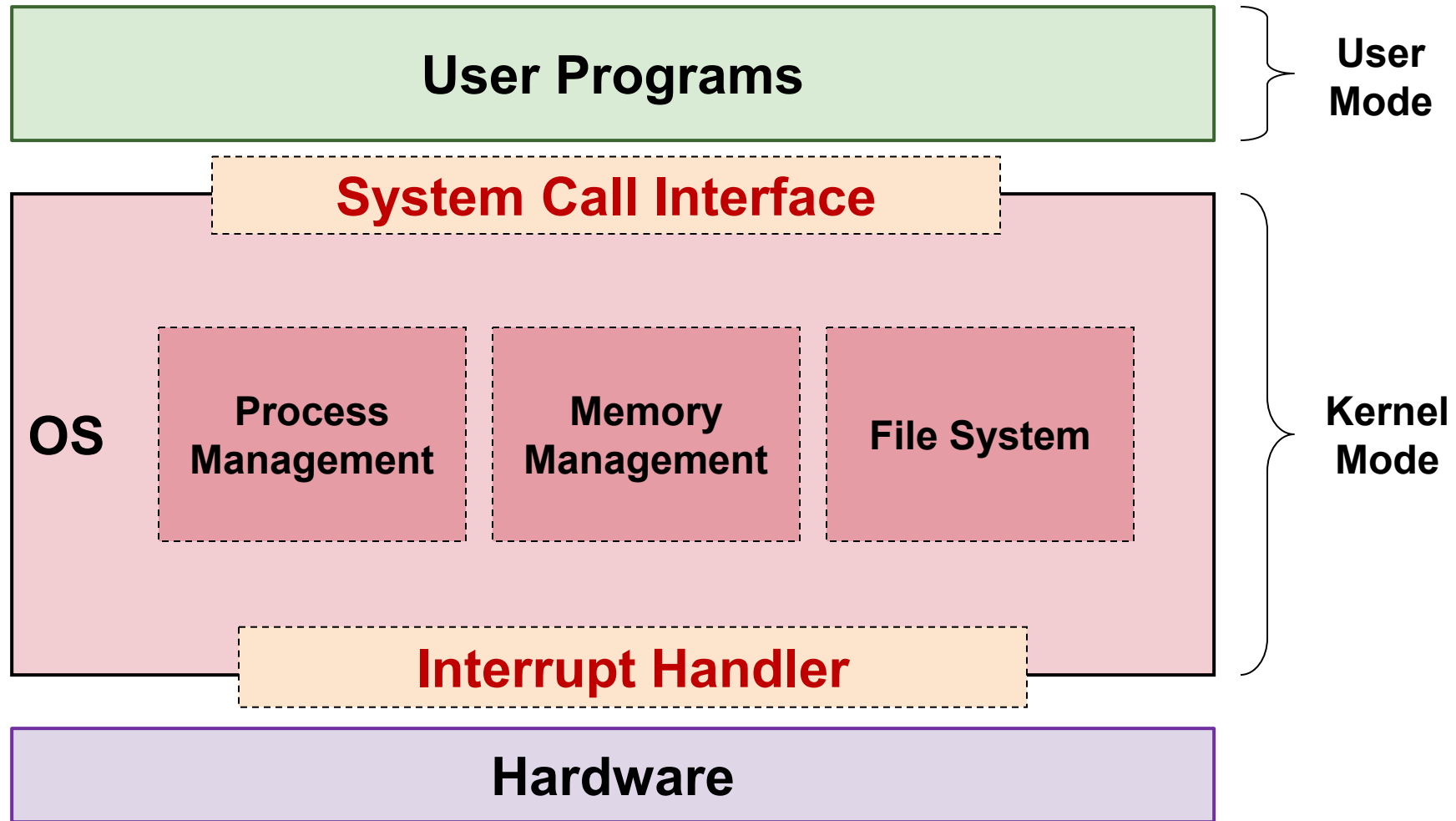
Concurrency

- Race conditions
 - Critical Section
 - Semaphore
 - Classical synchronization problems
-

Phew..... "quick summary" now

WHAT HAVE WE LEARNED?

Operating System



**WHAT ELSE
HAVE WE LEARNED?**

Side **Benefits**....

- Design of **complex system**
 - **Abstraction** and **Interface**
 - **Resource Management**
 - Performance Trade Off (**time** vs **space**)
-

What's **next**?

- **System Security**
 - **Parallel Computing/ Concurrent Programming**
 - **Computer Architecture**
 - **Compilers**
-

OH... THE **EXAM** 😊

The plan...

- Like the midterm
 - ❑ F2F in MPSH1 an MPSH2
 - ❑ Open book with printed materials
 - Backup in place:
 - ❑ LumiNUS quiz
 - ❑ Zoom proctoring
 - ❑ Record your screen
 - ❑ Refer to PDF materials
 - Email us early to book a consultation slot
-

Important to **know**

- **Rough percentage of coverage**
 - Lecture 1 to Lecture 5 = **~25%**
 - Lecture 6 to Lecture 11 = **~75%**
- **MCQ questions**
- **Short questions**
 - Write short answers
- **Open book**

It's Over!

Goodbye! Say Hi if you see us in school!