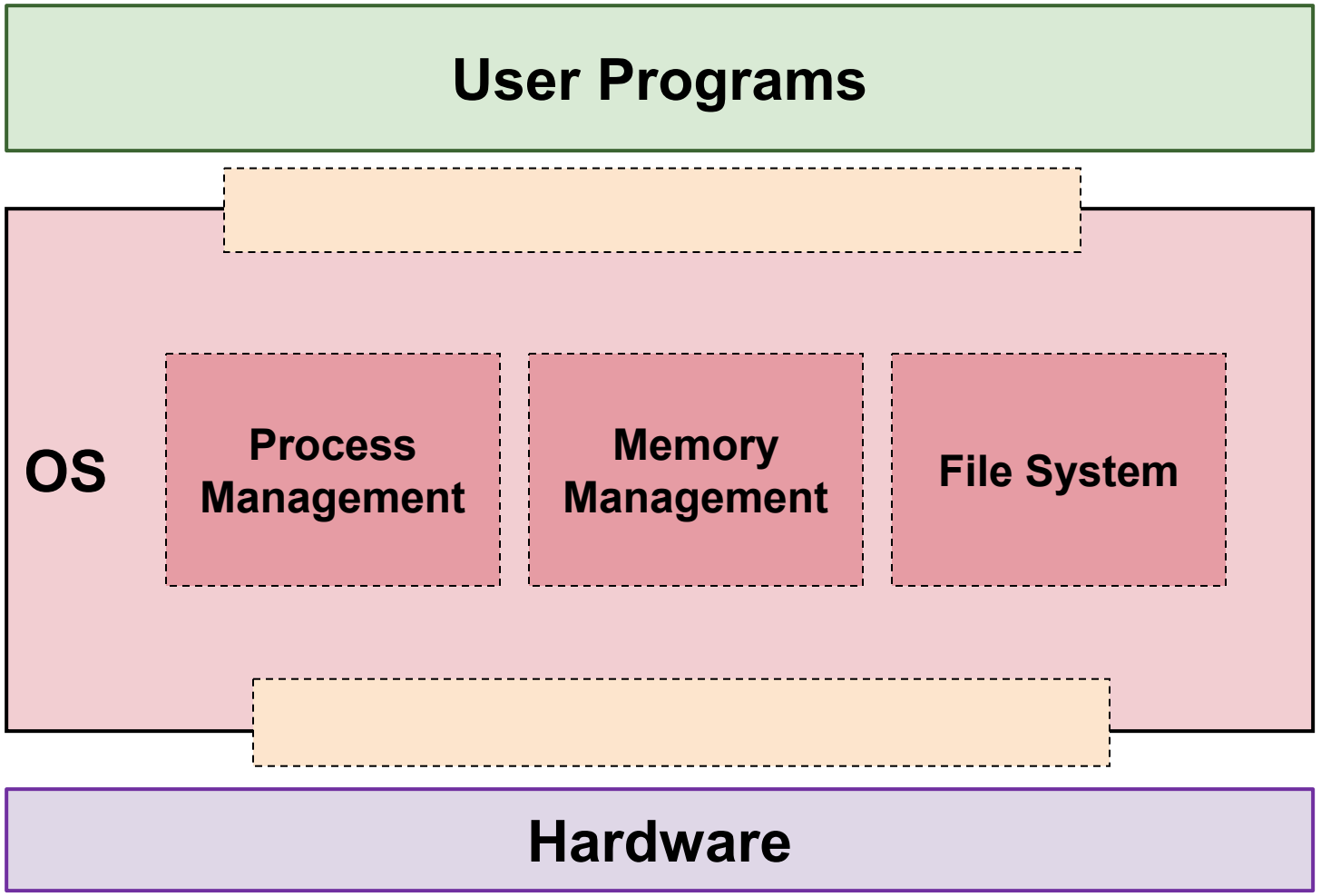
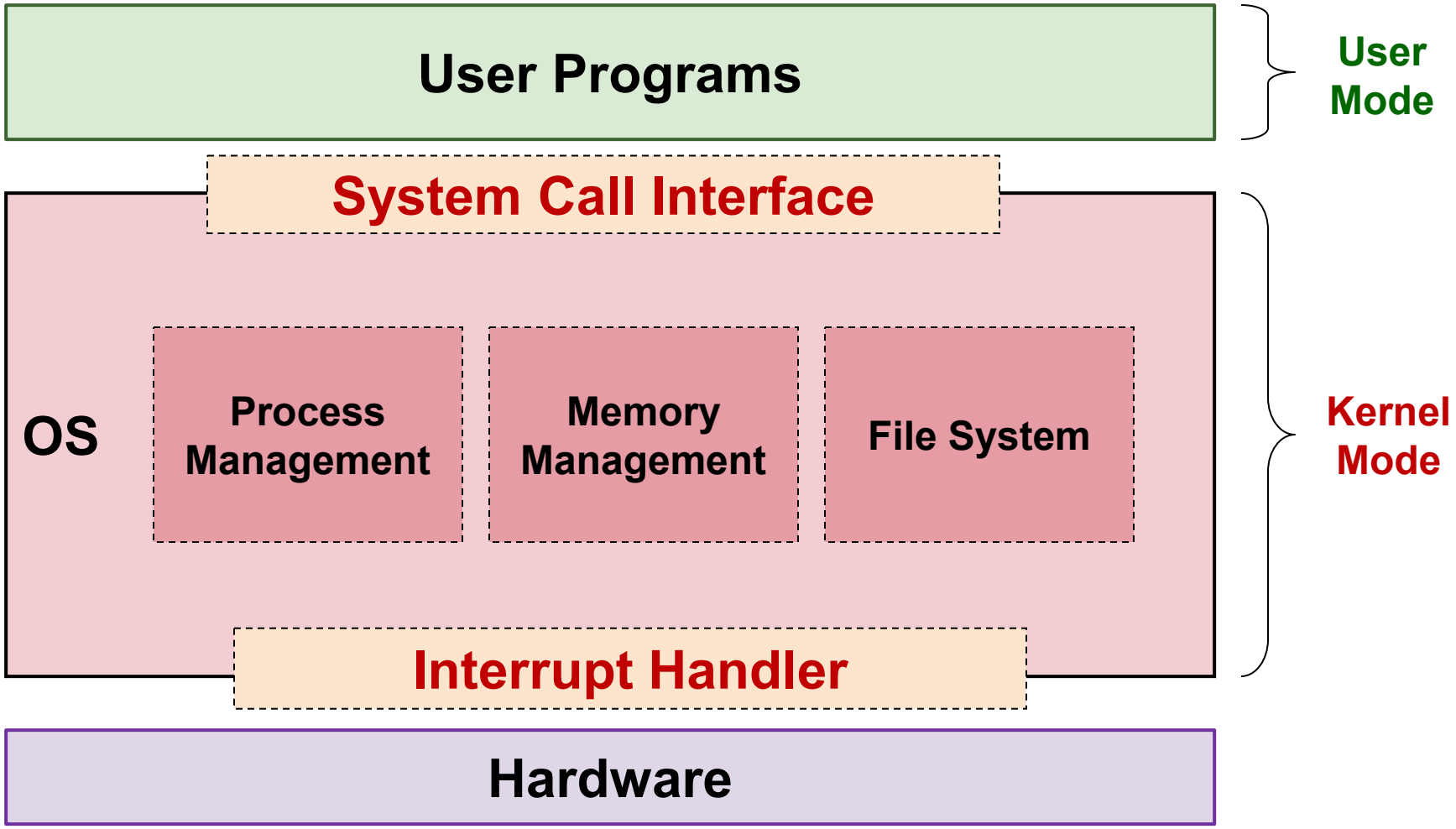
# Big Bang & Revision

Lecture 12 (**Live Version**)

# OS Structure (Monolithic)



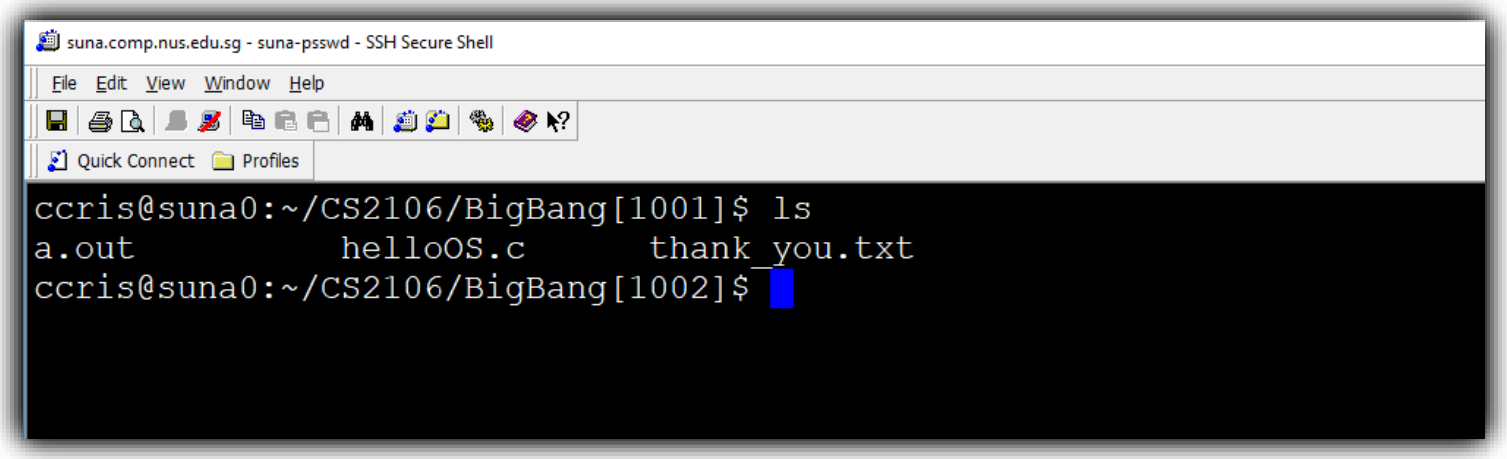
# OS Structure (Monolithic)



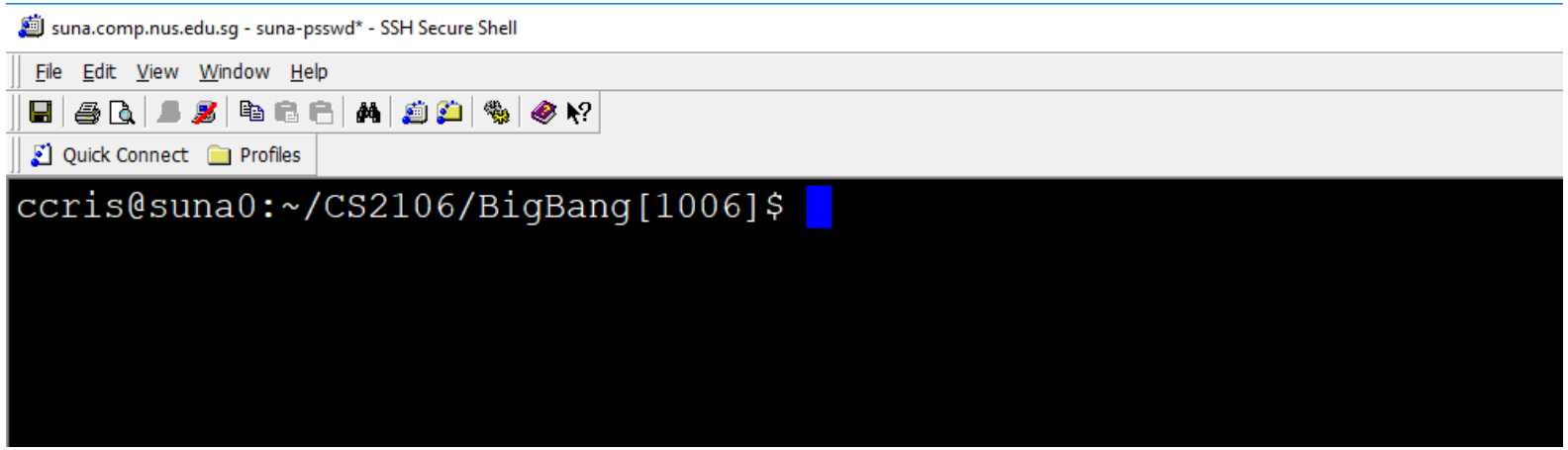
*23 + 11 + **lots of** hours just to learn about what happened in **1 second***

# THE STORY OF
# SHELL INTERPRETER

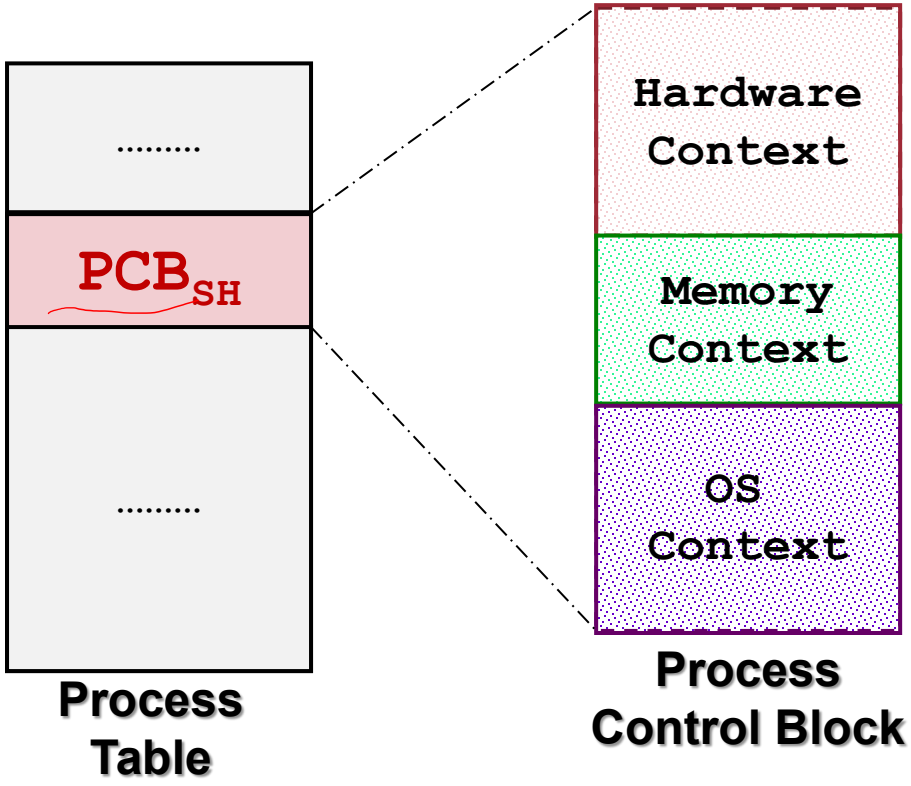
# Ah…. Ah…. so simple….. **or is it**?



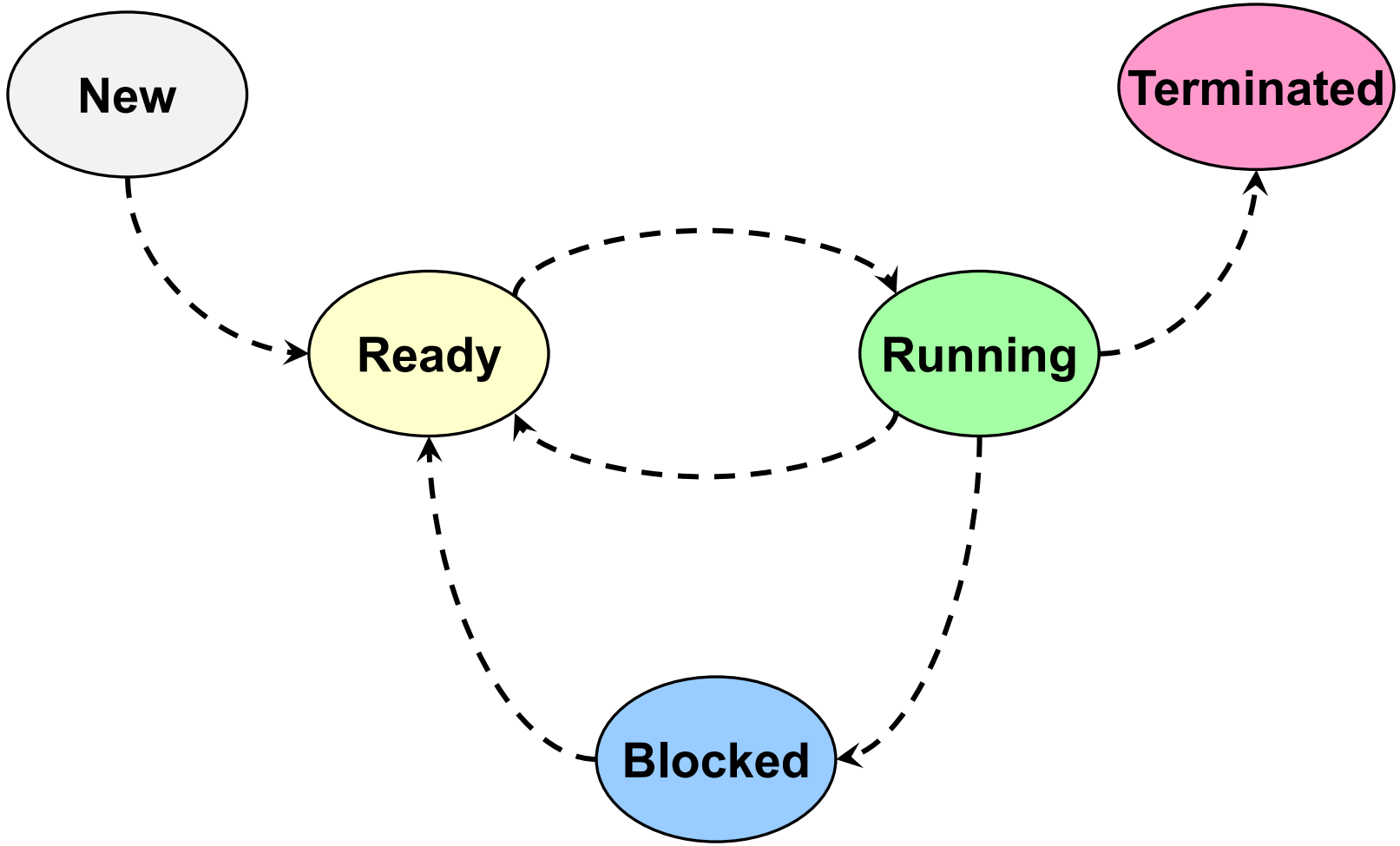
# Here we go … …


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

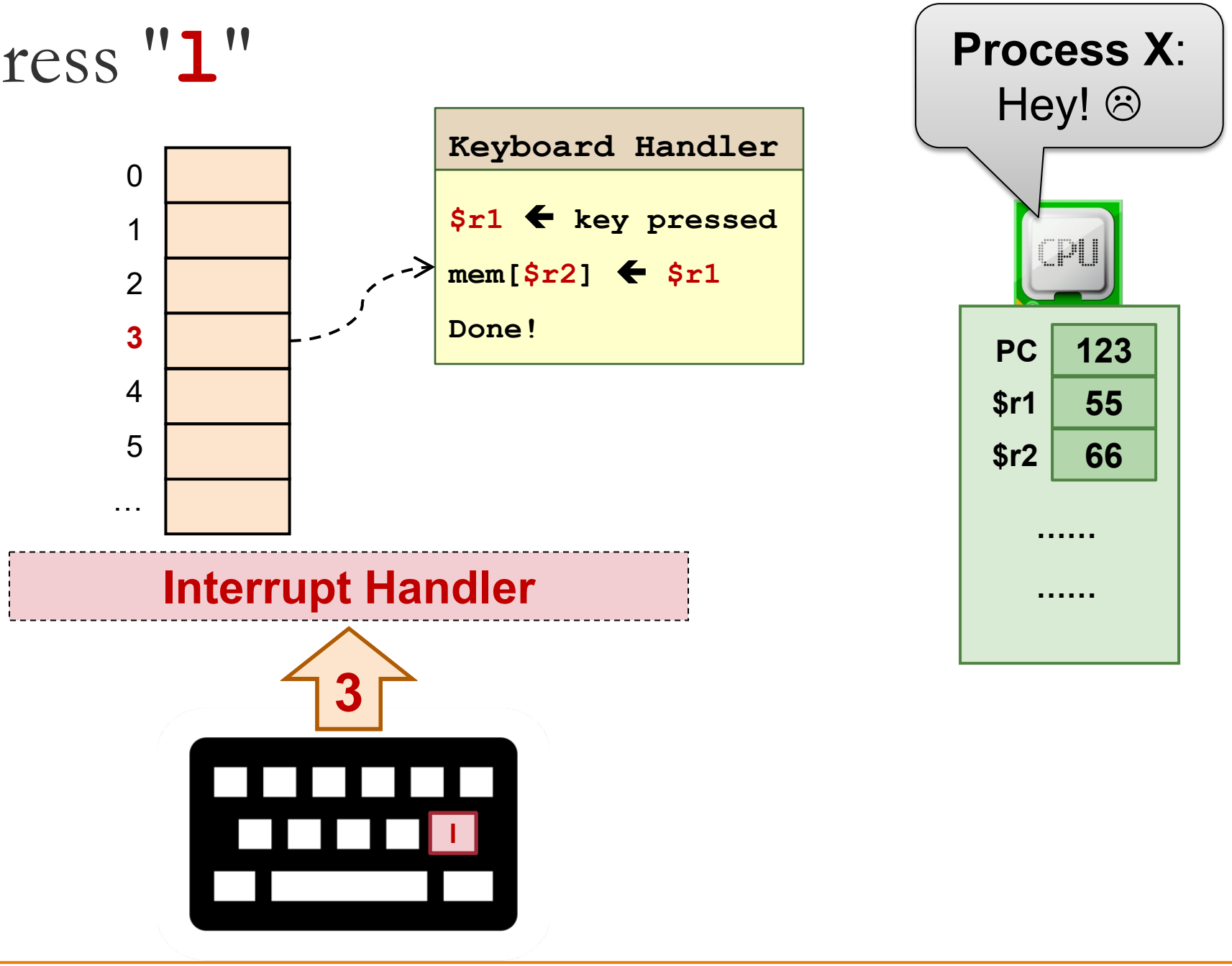
# Process Table & Process Control Block



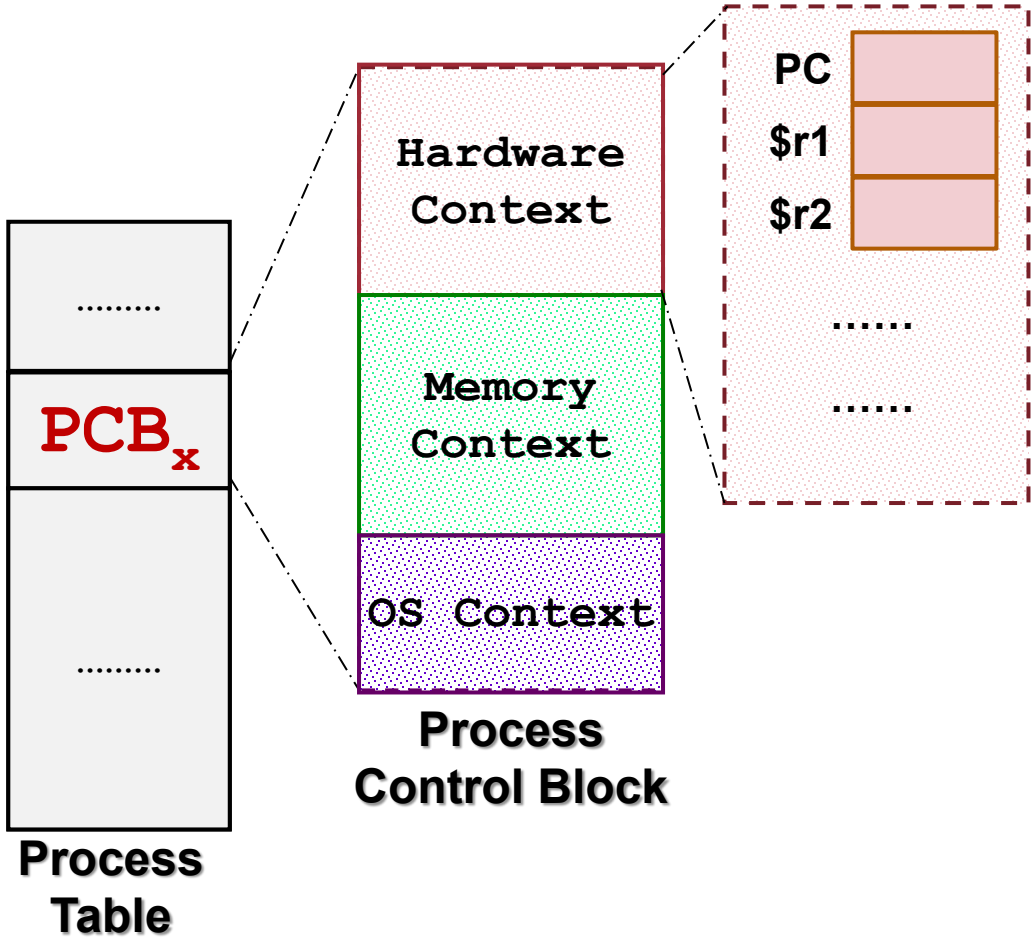
# The Interpreter is in…..

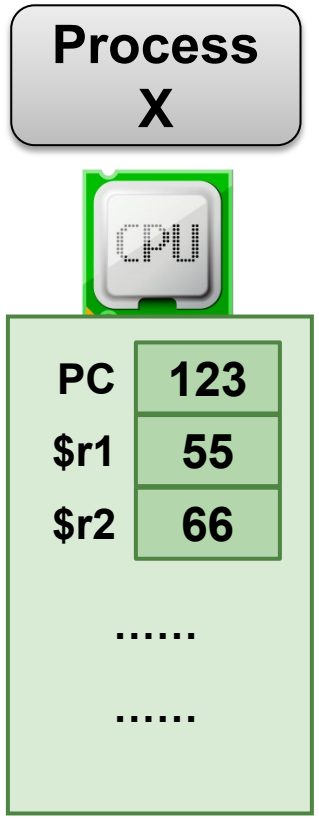


# User press "1"


0
1
2
3
4
5
...
Keyboard Handler
$r1 ← key pressed
mem[$r2] ← $r1
Done!
Process X:
Hey! ☹
CPU
PC 123
$r1 55
$r2 66
.....
.....
Interrupt Handler
3
1

# Sorry to **interrupt** you….



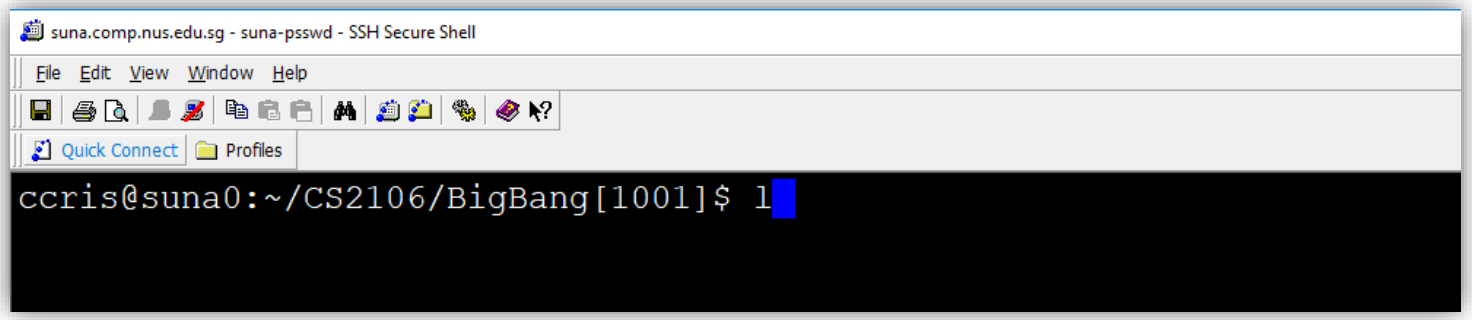

**Keyboard Handler**

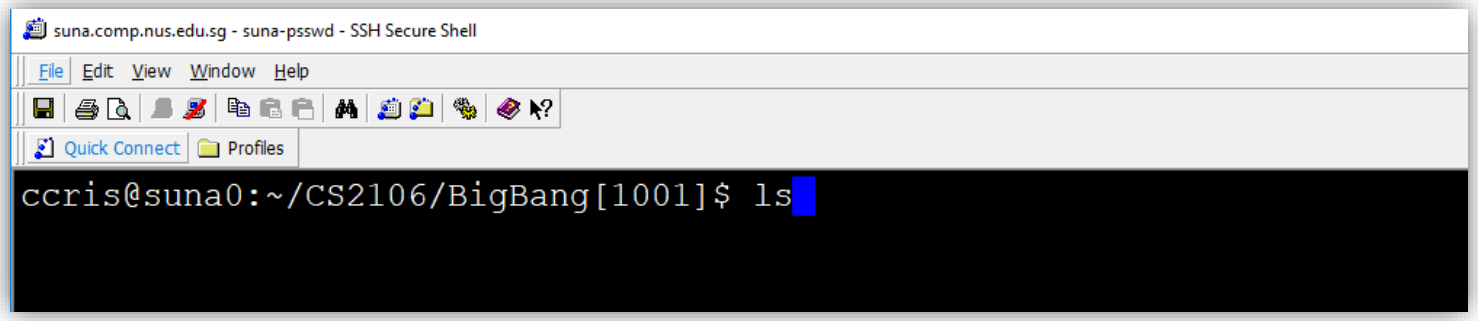
```
$r1 ← key pressed
mem[$r2] ← $r1
Done!
```

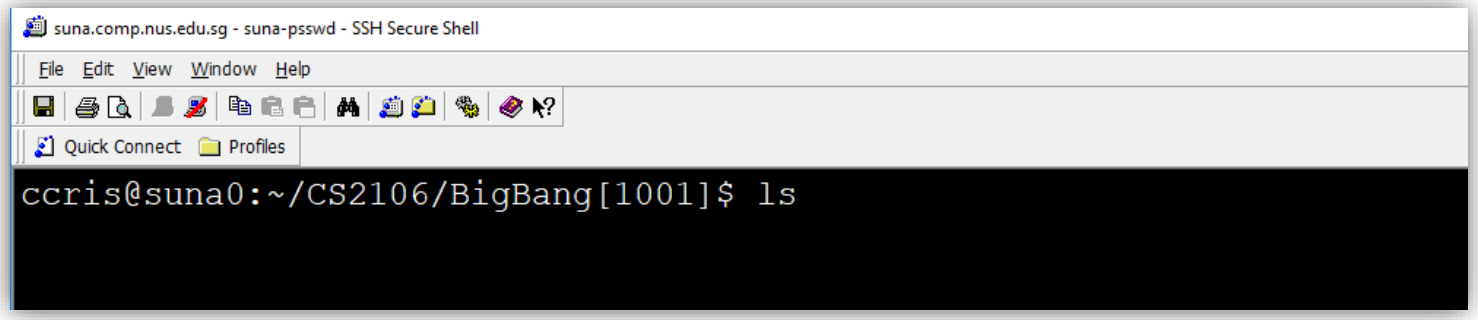
# Interrupt steps

- 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

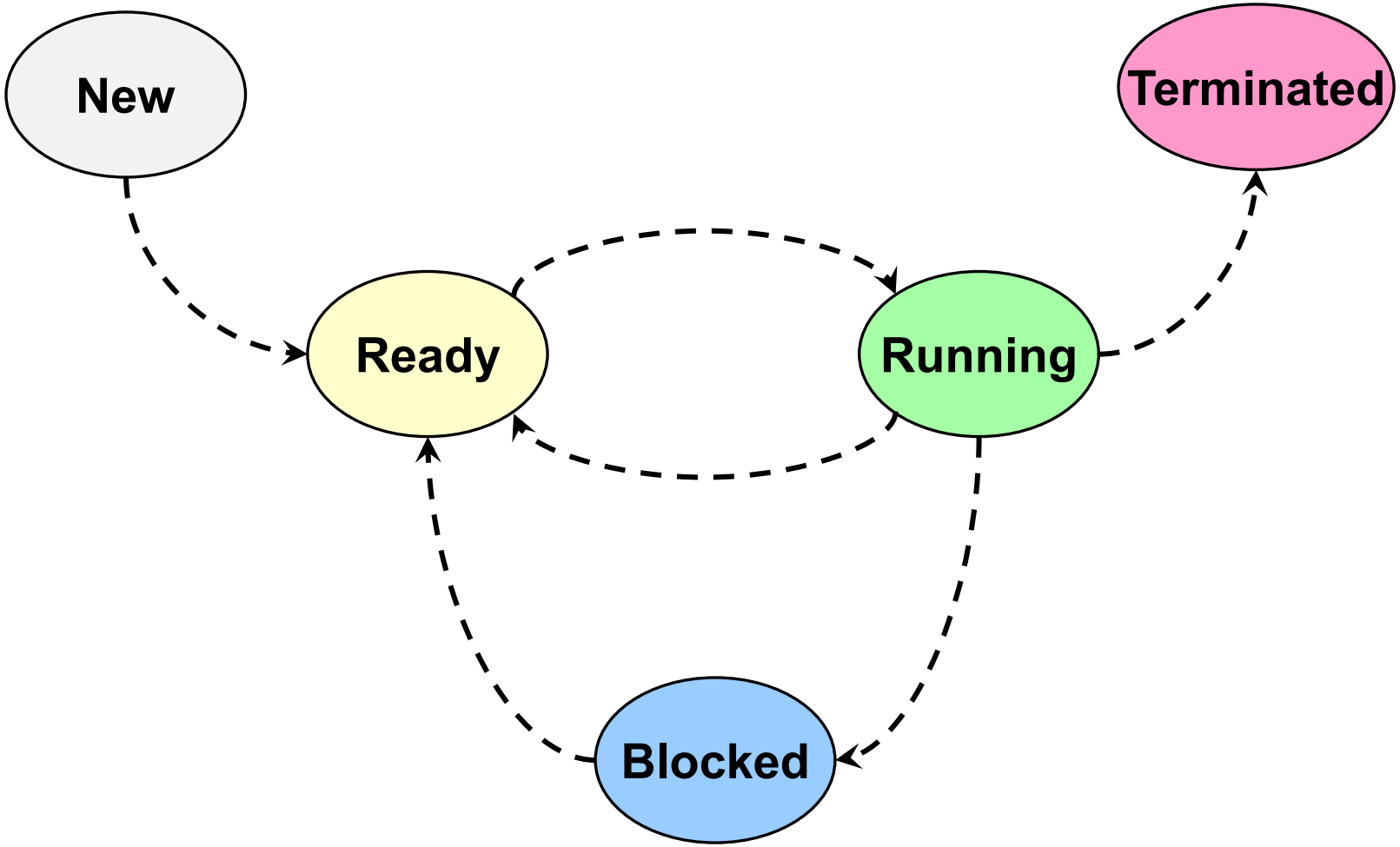
# Rinse and Repeat……



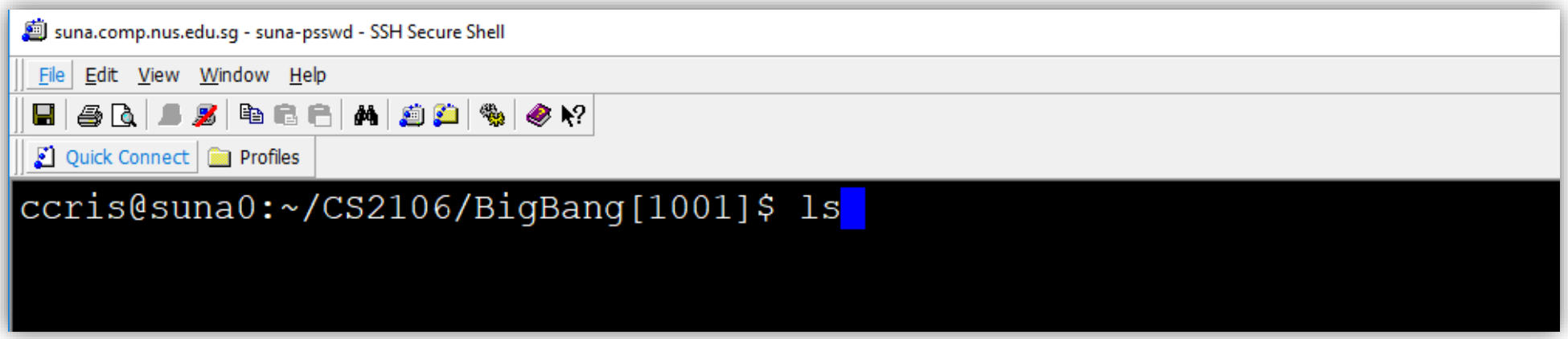



# The **interpreter** is now…



# User entered "ls", the interpreter will…




ls entered

# 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

RegisterX

```
int fork()
{




}
...
int main()
{
    ...
    fork();
    ...
}
```

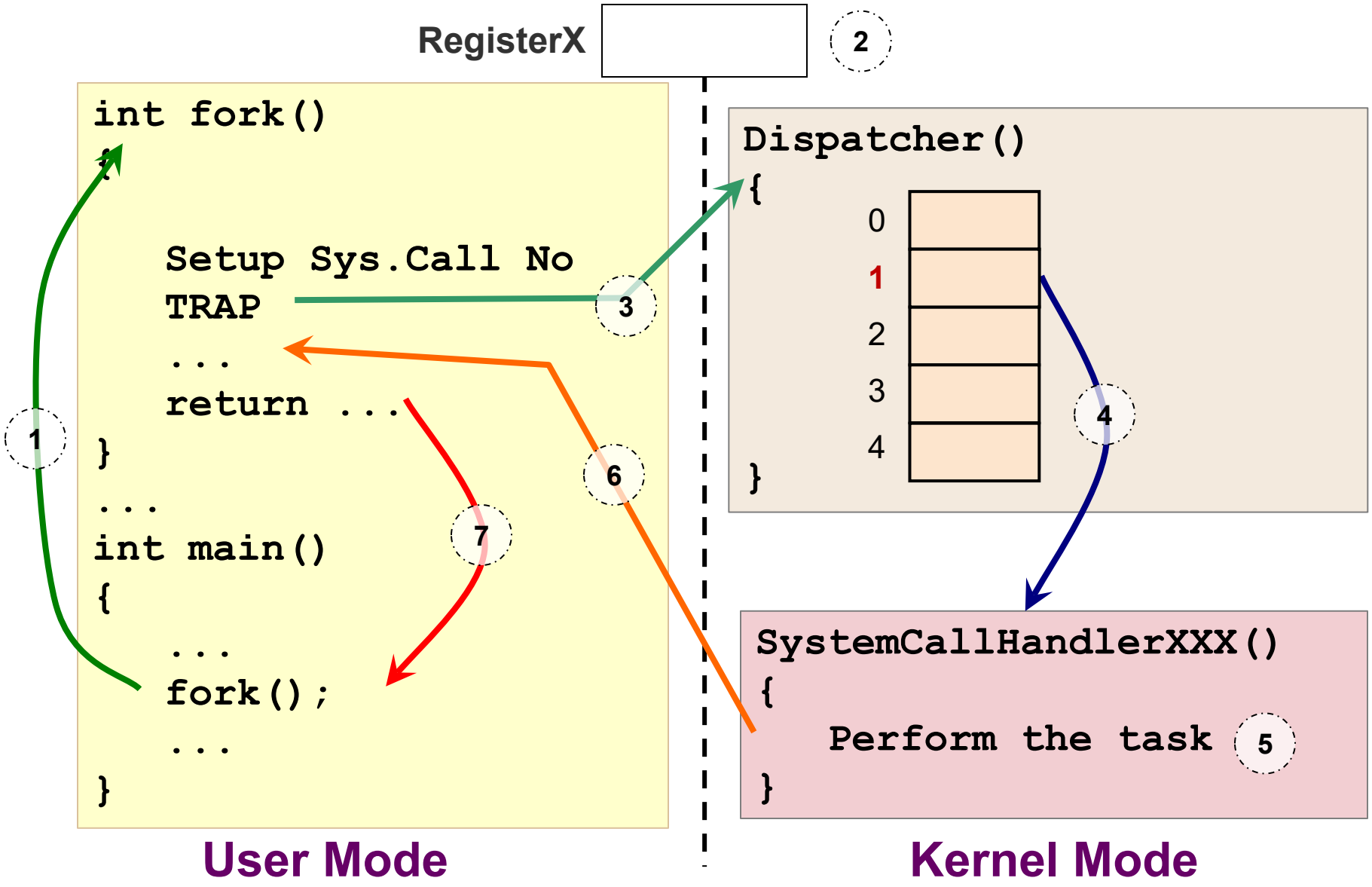
User Mode

```
Dispatcher()
{




}
```

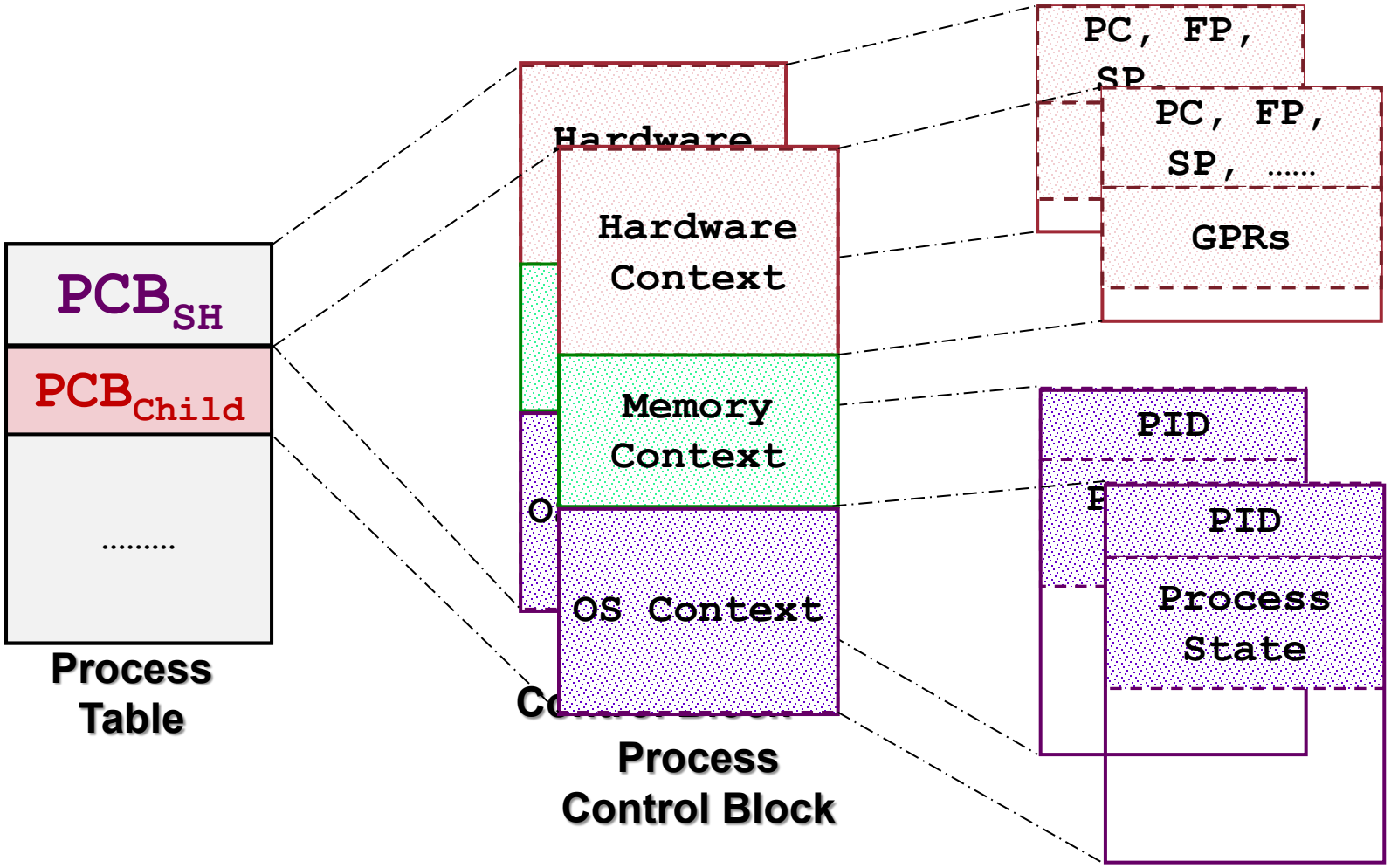
```
SystemCallHandlerXXX()
{

}
```

Kernel Mode

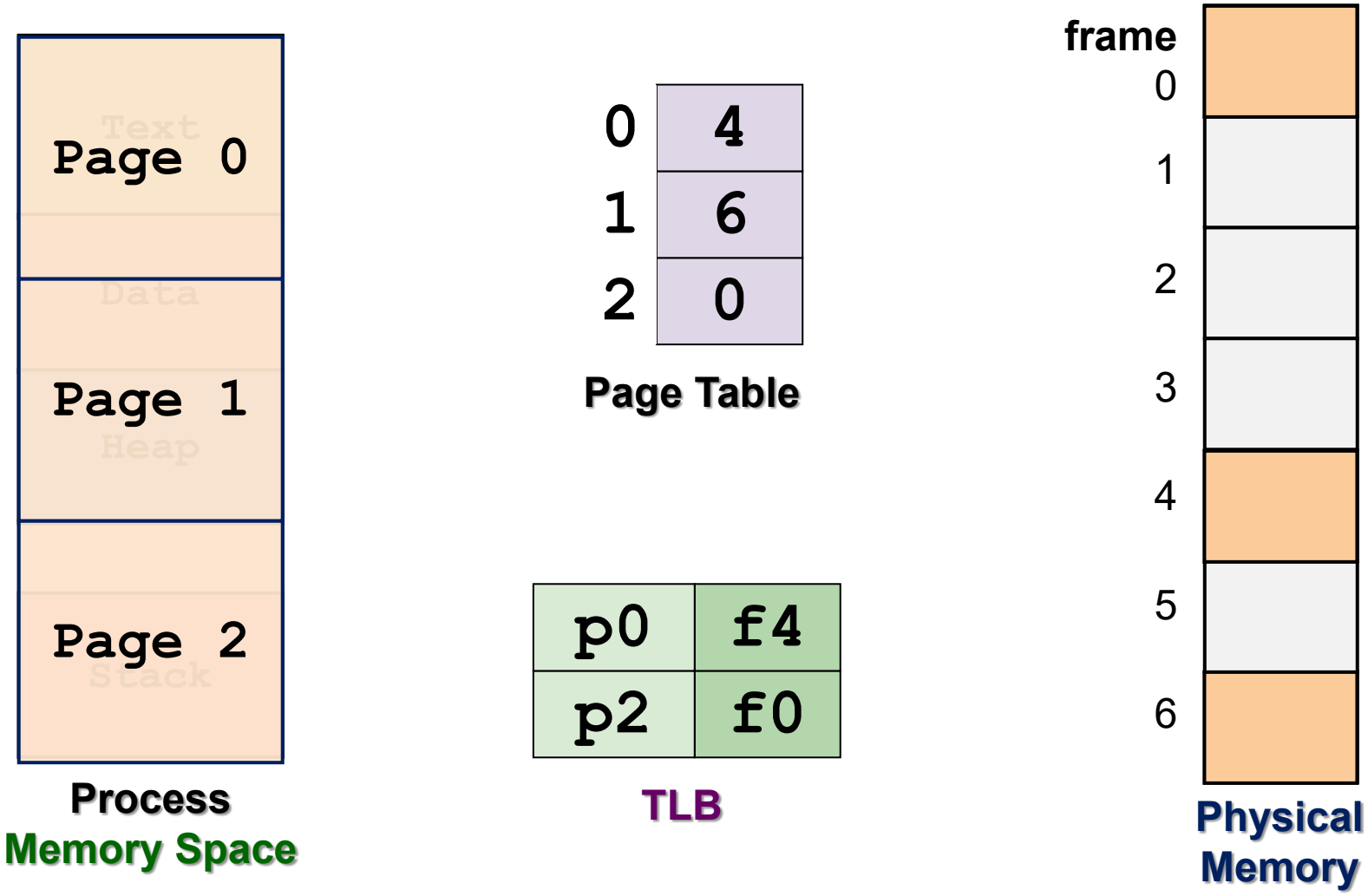
# fork() involves a system call



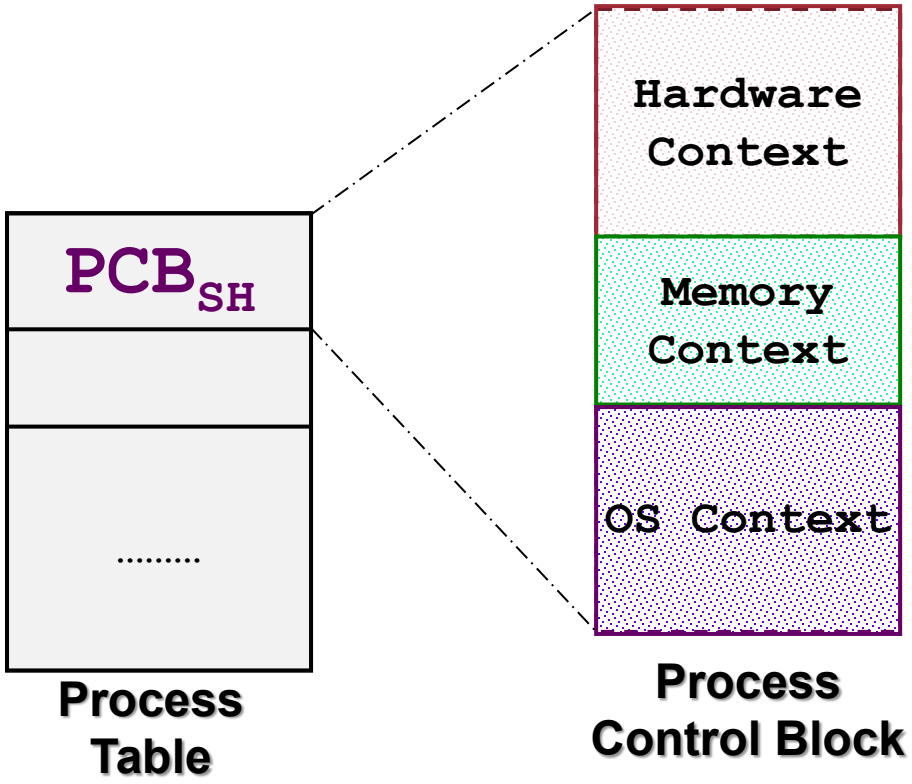
# What is the **effect** of fork( )?



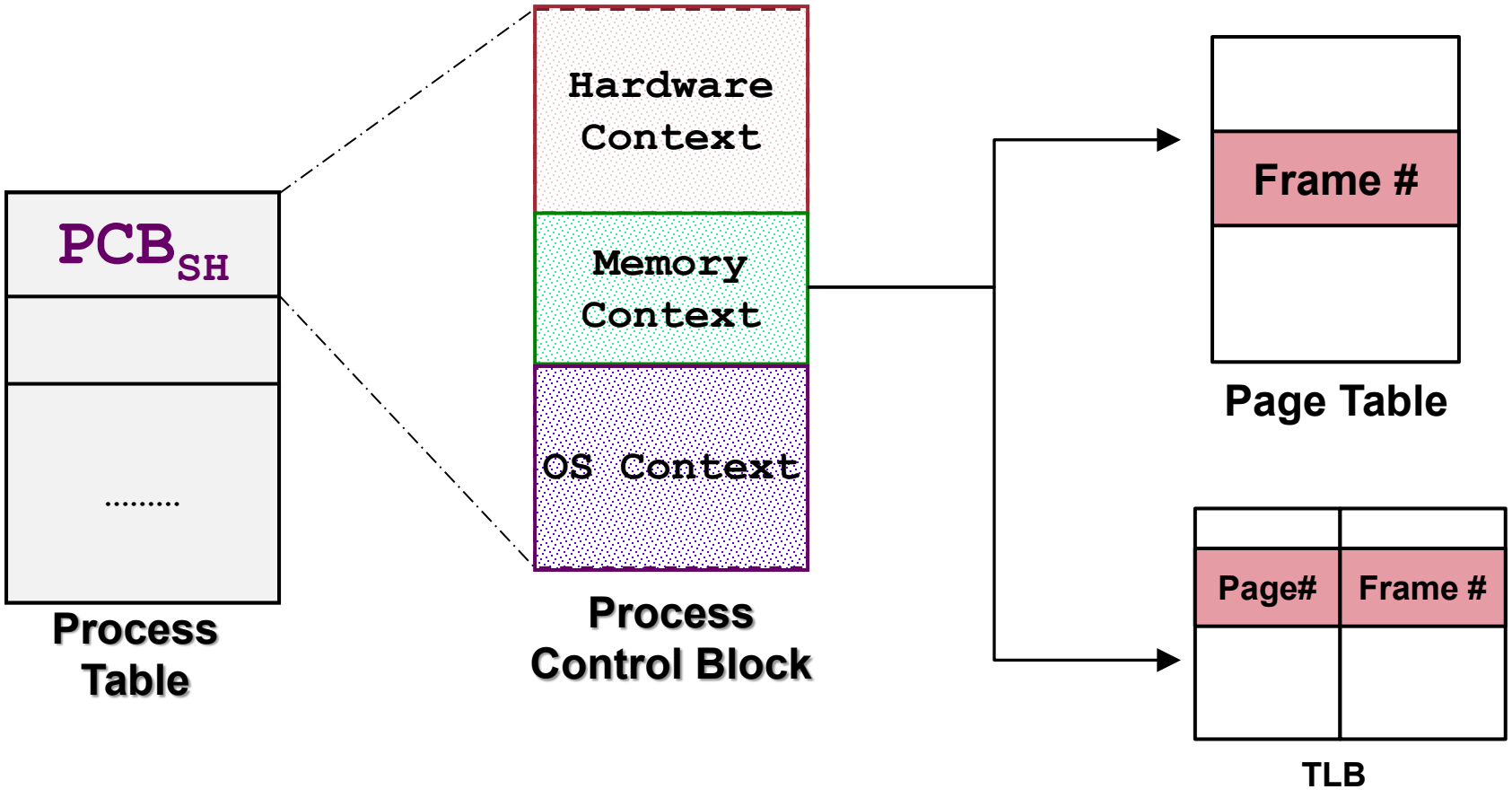
# Memory Space of a Process



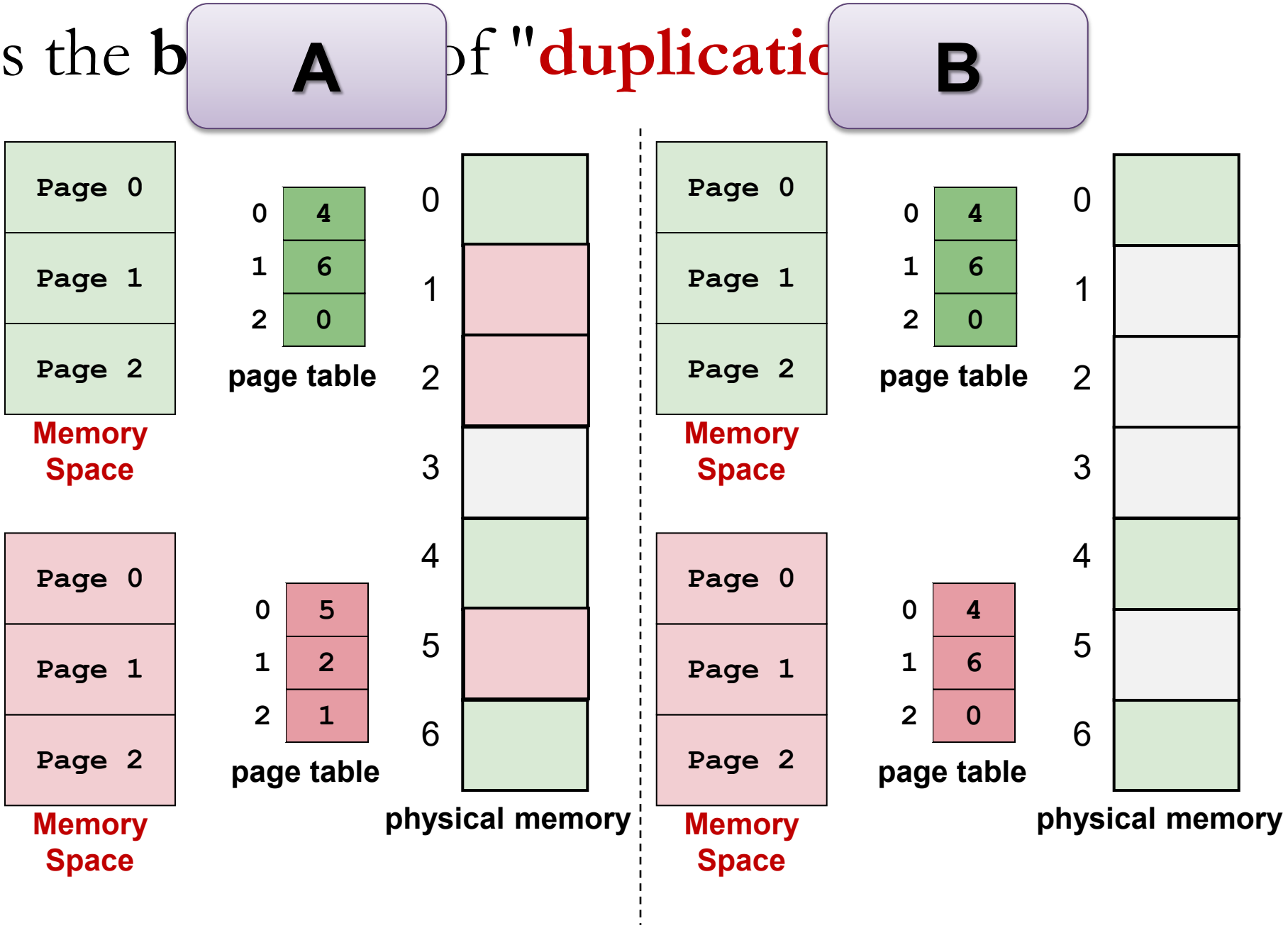
# Memory Context = ?



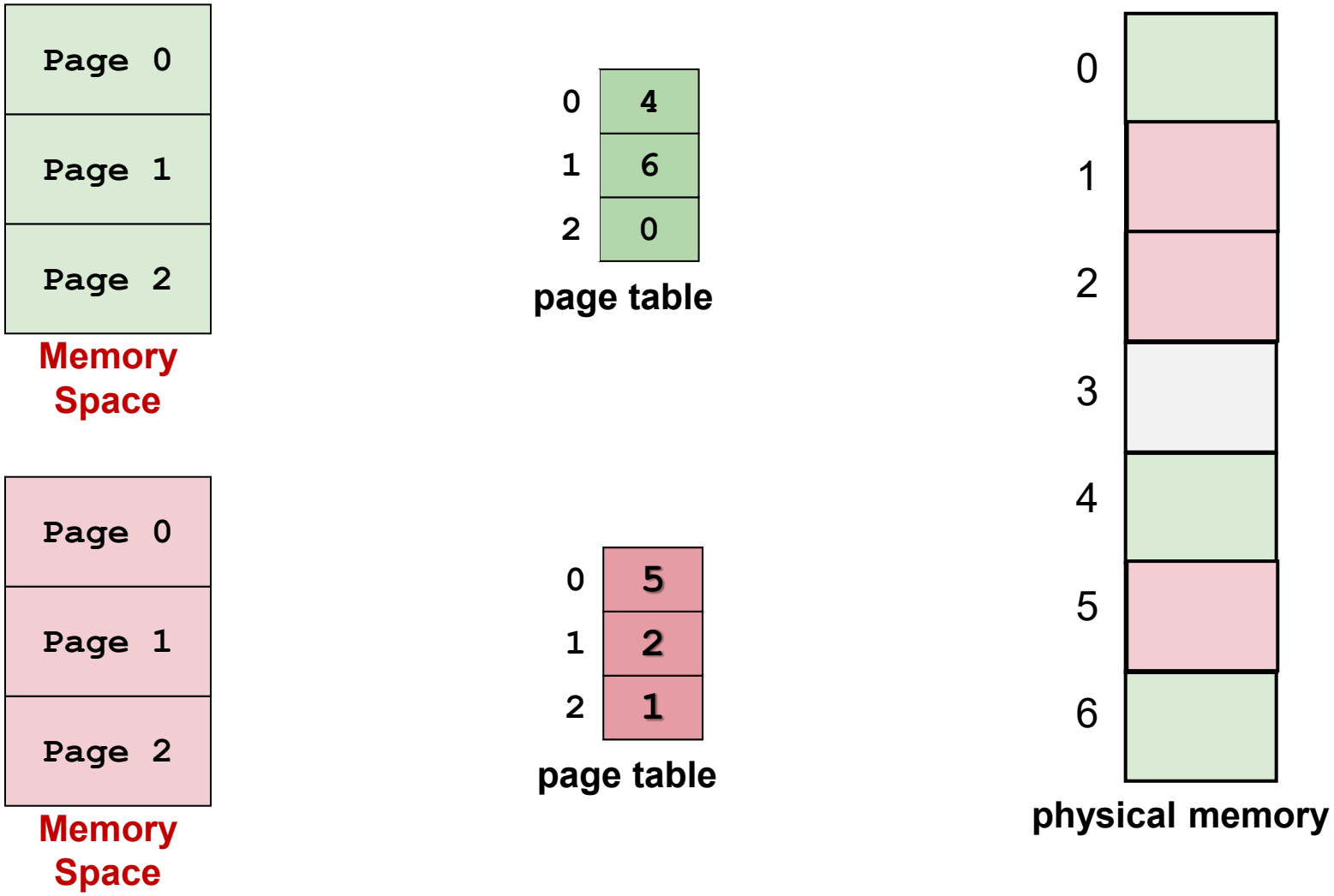
# Memory Context = ?



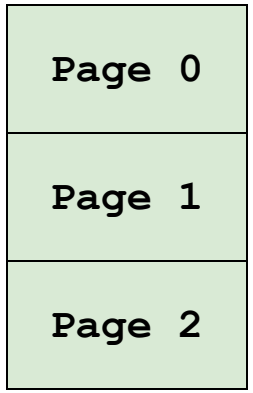
# What is the b of "duplicatio



# Duplicating Memory Space the HARD WAY



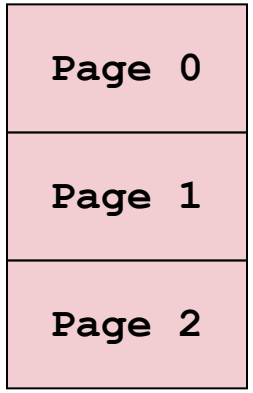
# Copy on Write




Memory Space

| | |
|---|---|
| 0 | 4 |
| 1 | 6 |
| 2 | 0 |

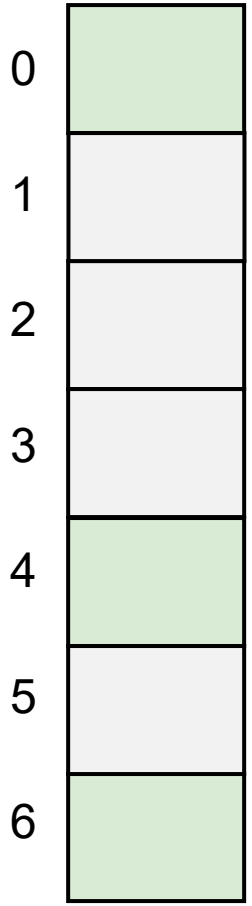
page table




Memory Space

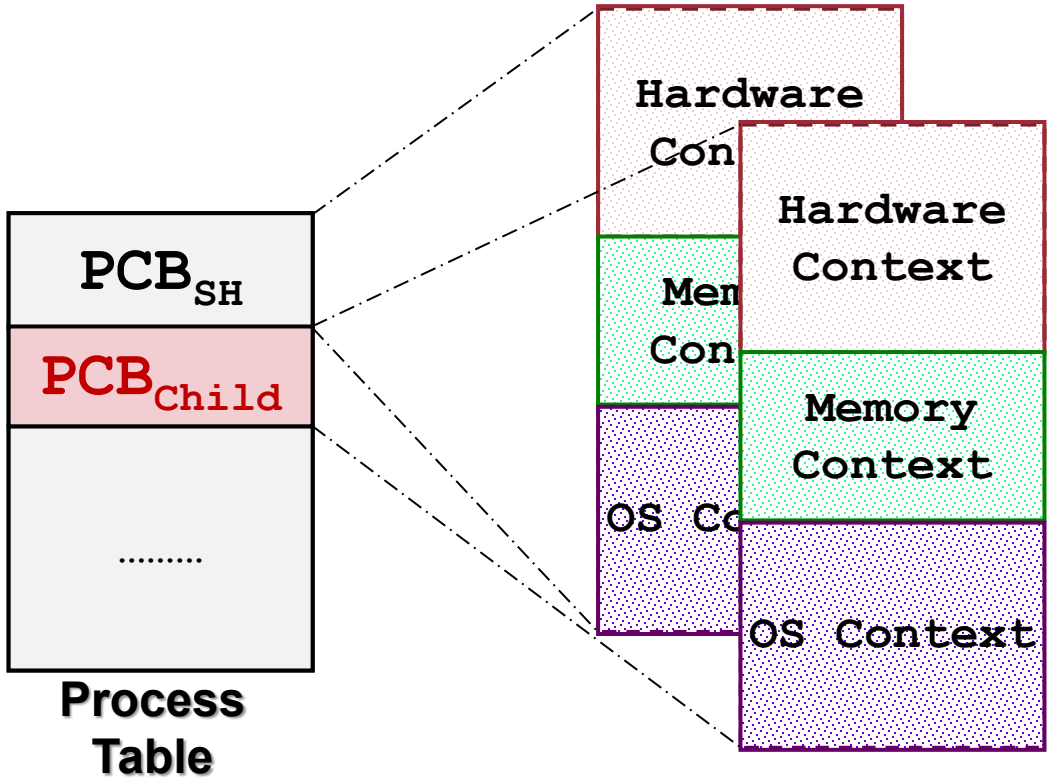
| | |
|---|---|
| 0 | 4 |
| 1 | 6 |
| 2 | 0 |

page table

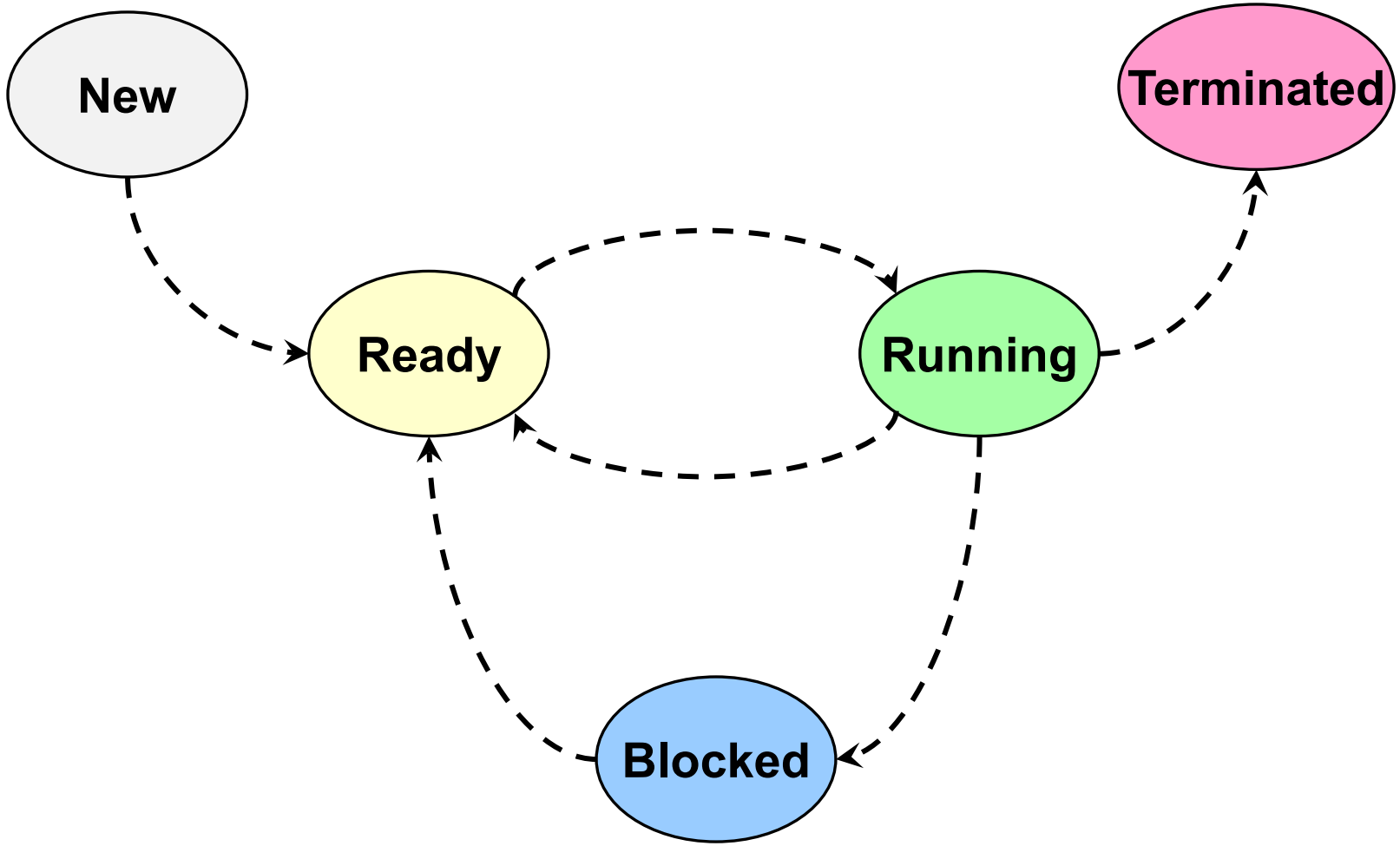



physical memory

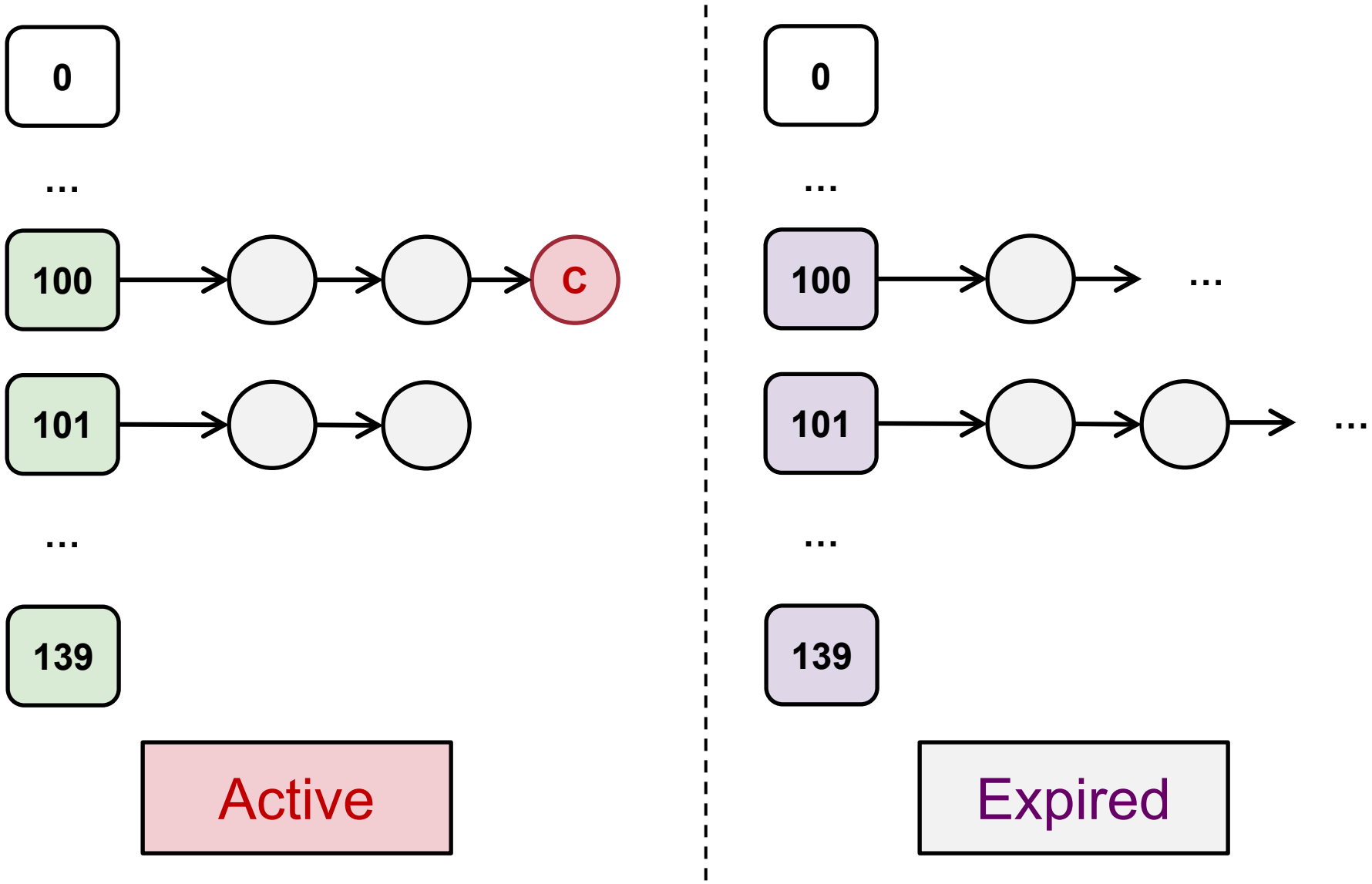
# So, Effect of **fork**( )



# **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 | |

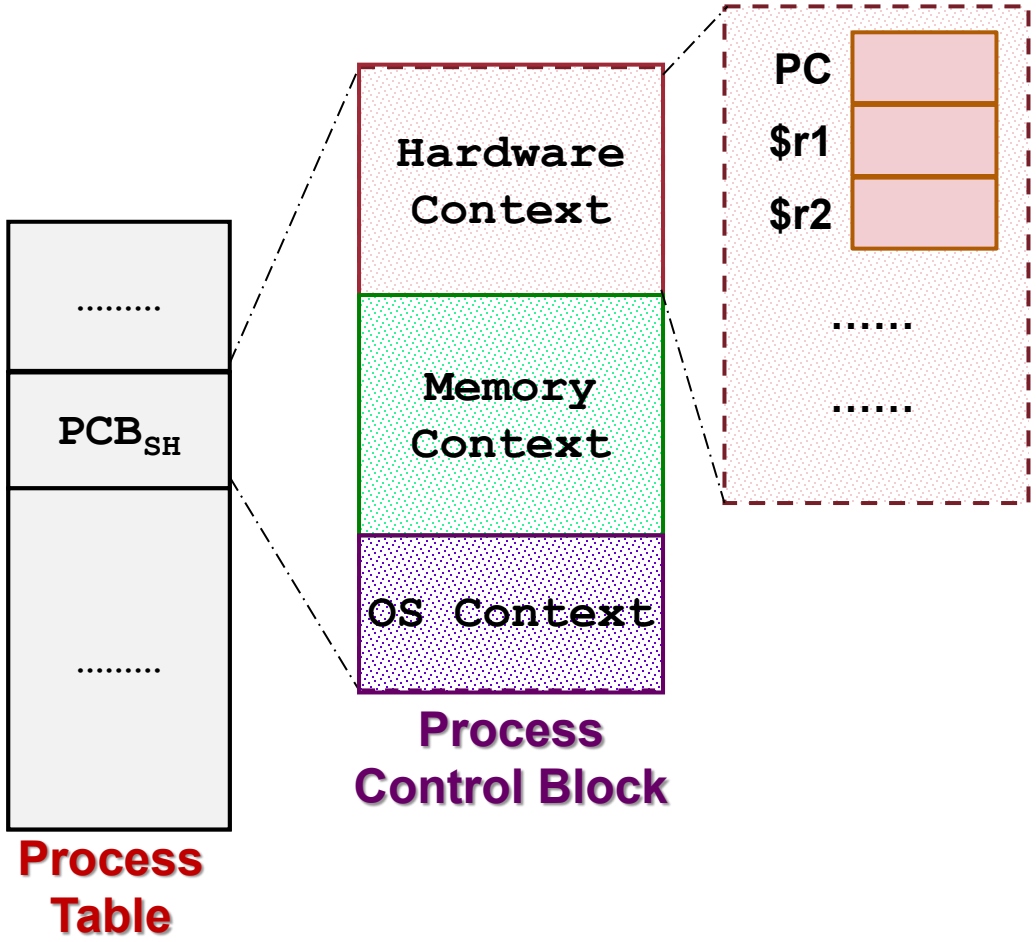
```
SystemCallHandlerXXX()
{
    Perform the task
}
```

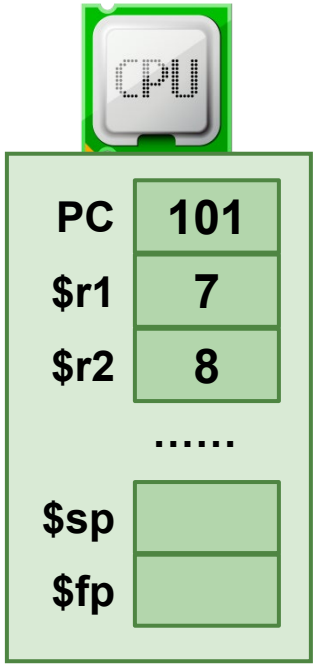
**Kernel Mode**

# 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….

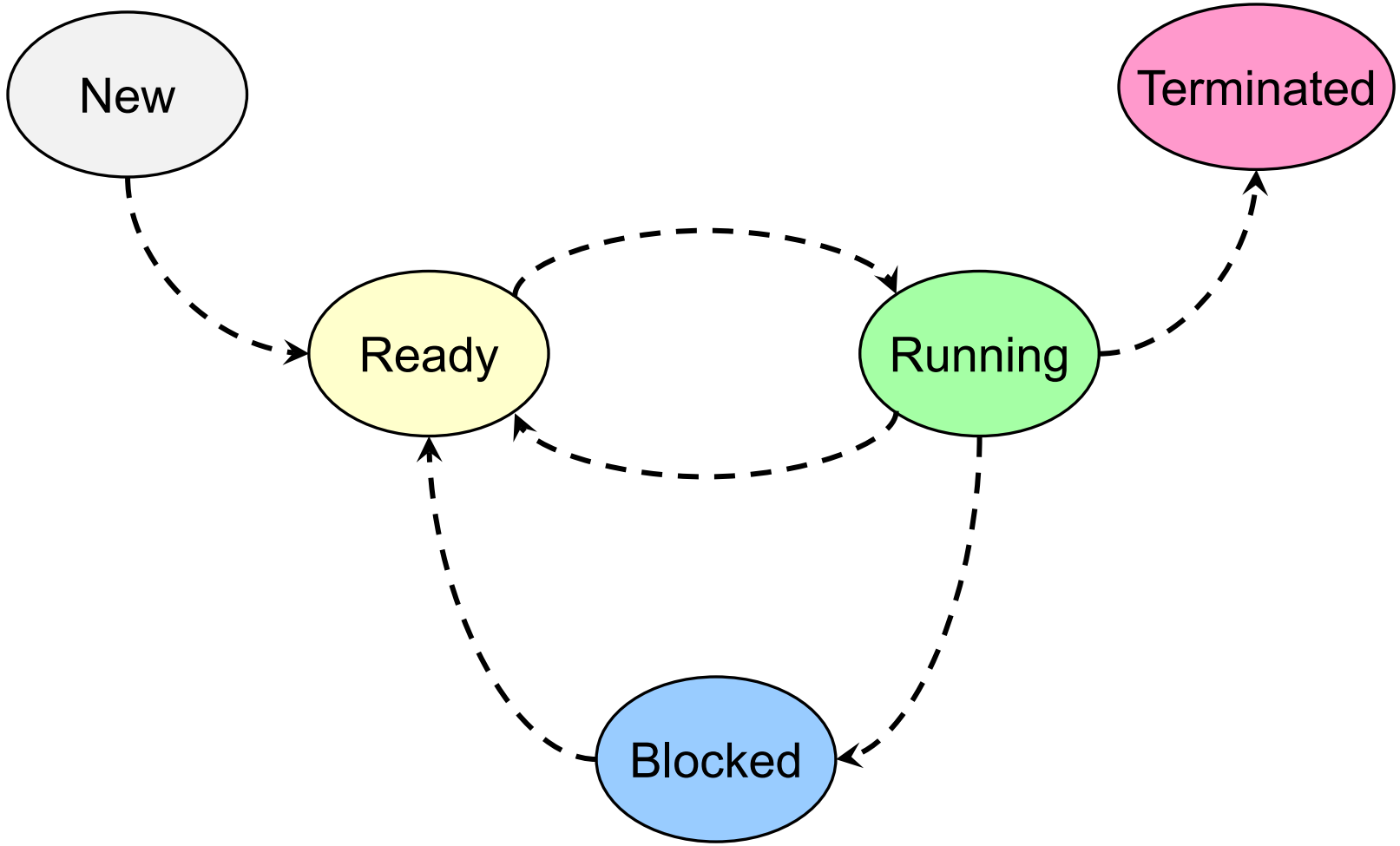


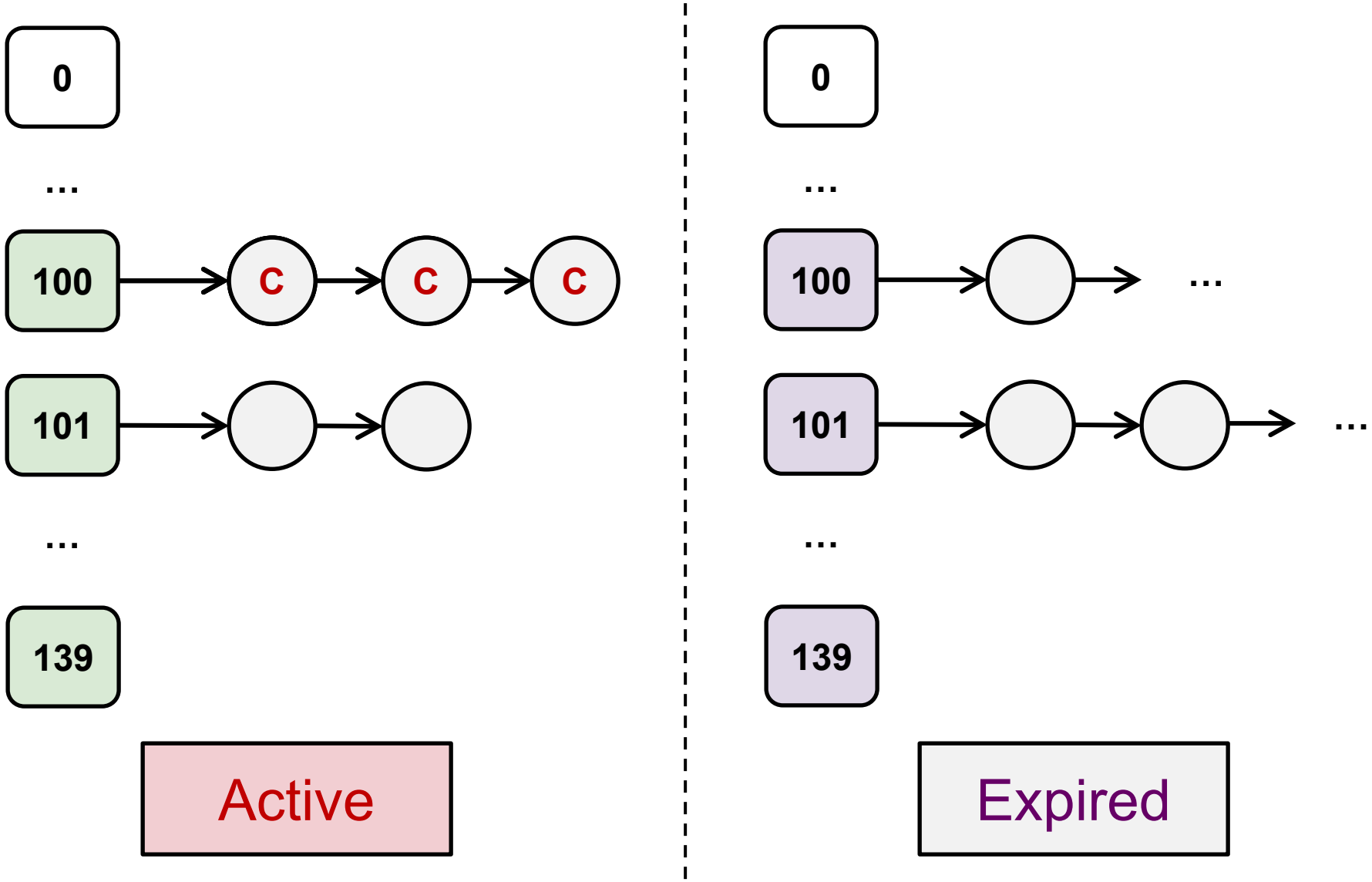

| | |
|---|---|
| p0 | f4 |
| p2 | f0 |

TLB

# The interpreter is now…

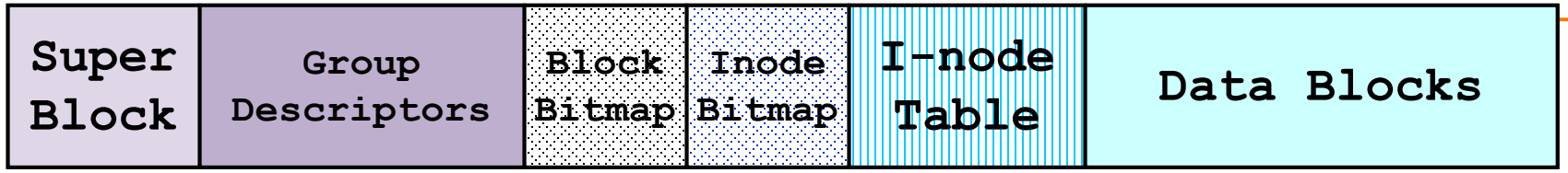


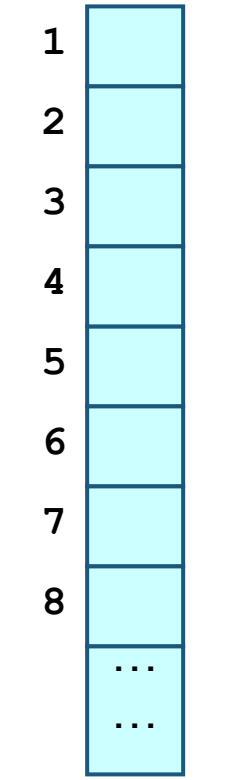
# Hmm… **who** gets to **run**?



# Typical steps for Shell Interpreter

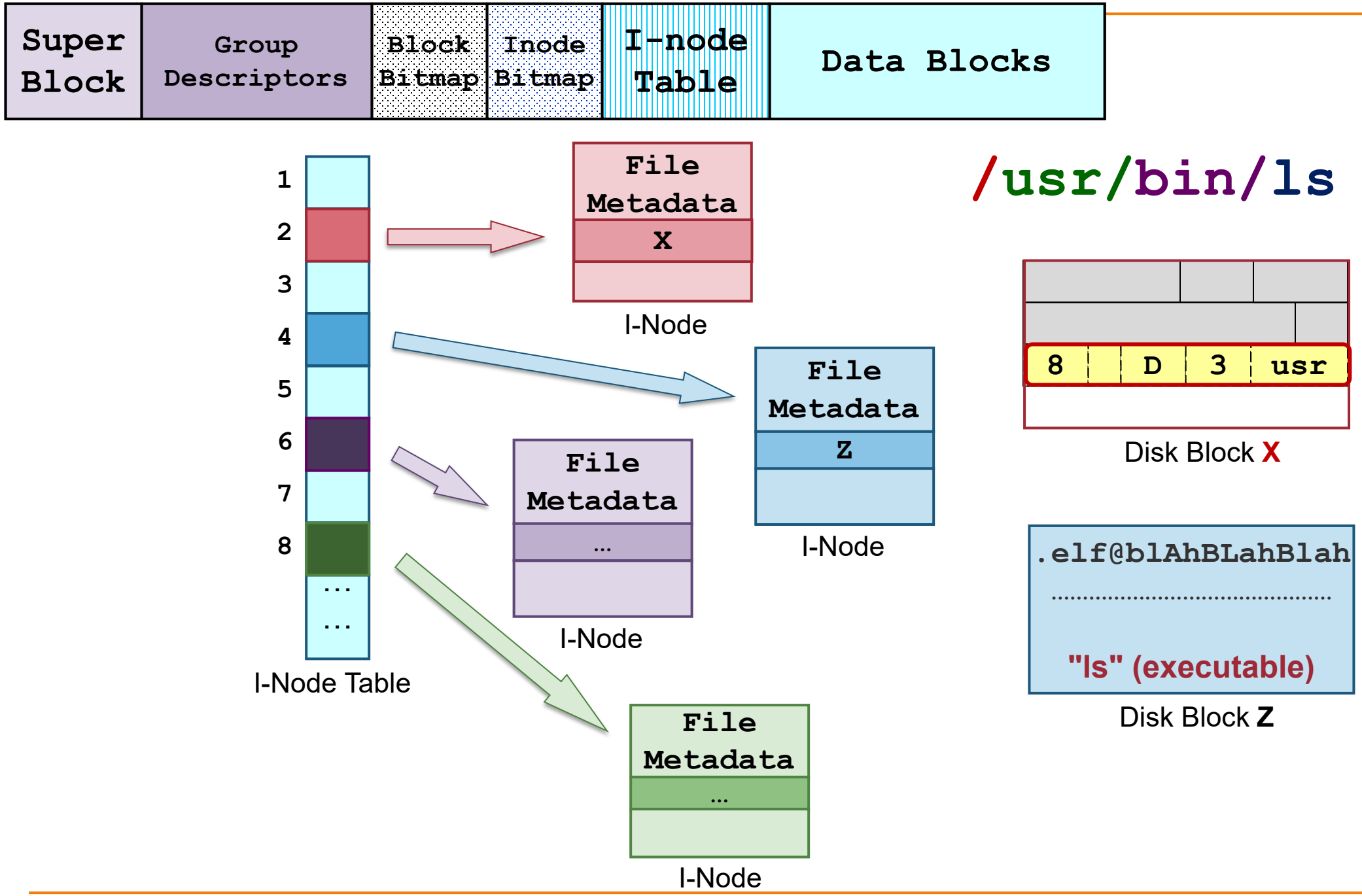
```
UserCmd  ← Read from keyboard
fork()
if I am the parent (i.e. the shell)
    wait ( child to finish )
else
    exec( UserCmd )
```



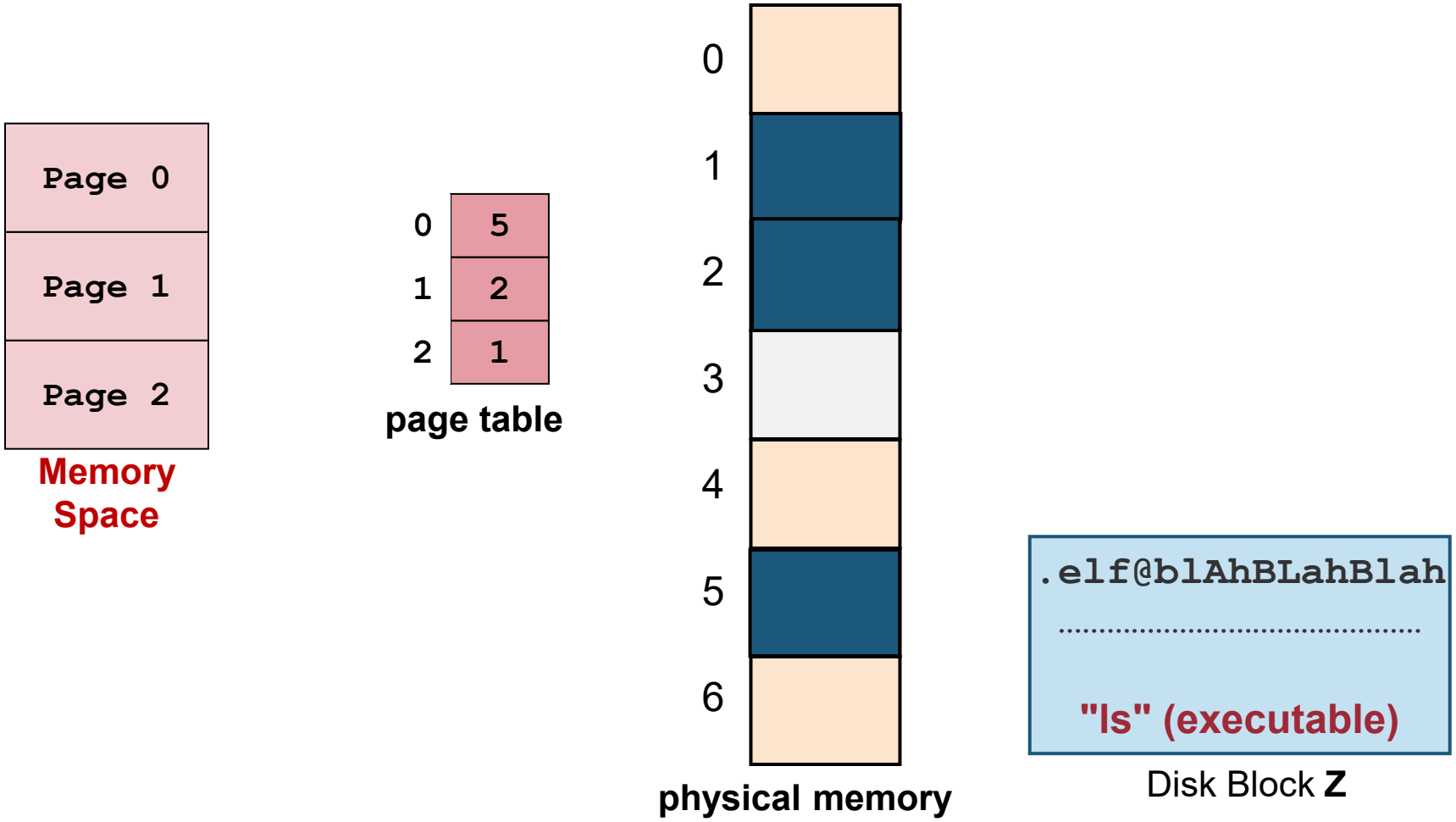




I-Node Table

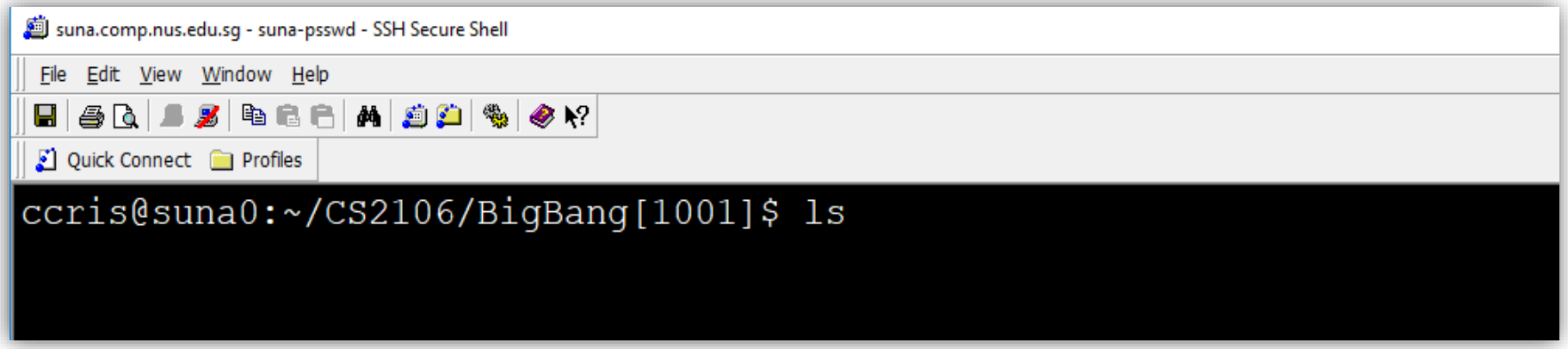
/usr/bin/ls


Super Block
Group Descriptors
Block Bitmap
Inode Bitmap
I-node Table
Data Blocks
1
2
3
4
5
6
7
8
...
...
I-Node Table
File Metadata
X
I-Node
File Metadata
Z
I-Node
File Metadata
...
I-Node
File Metadata
...
I-Node
/usr/bin/ls
8
D
3
usr
Disk Block X
.elf@blAhBLahBlah
"ls" (executable)
Disk Block Z

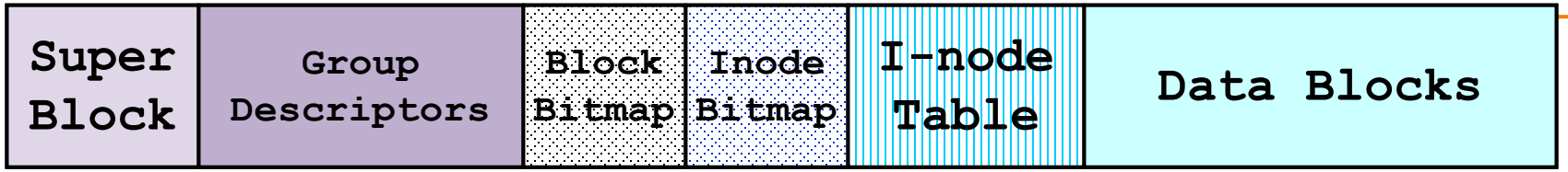
# Memory Content Replaced



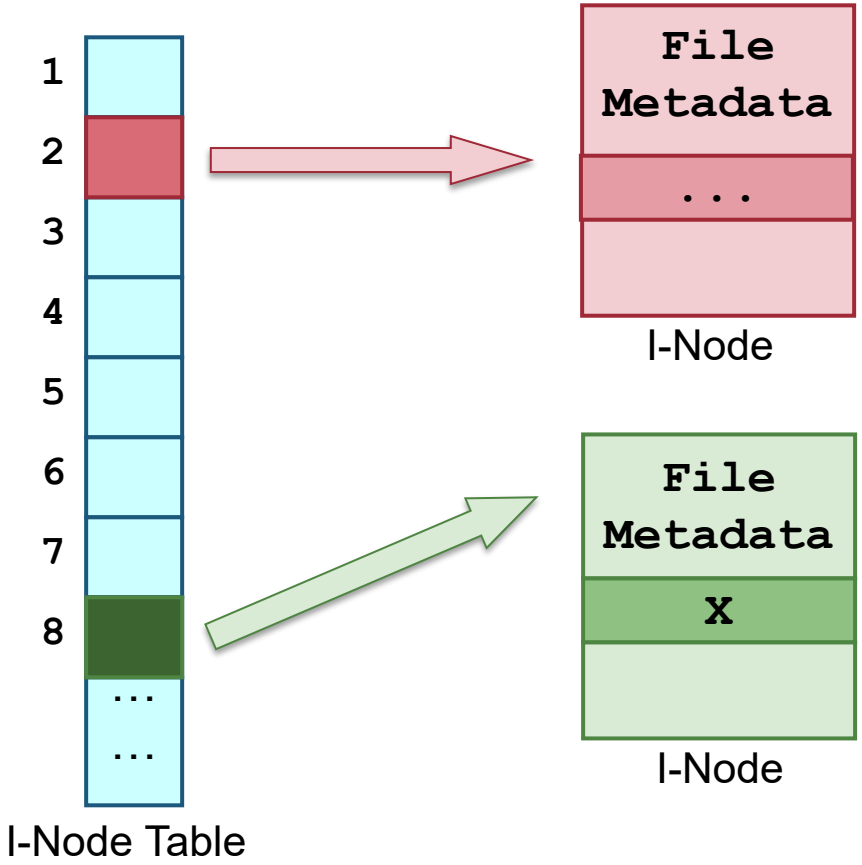
# Child is now "**ls**", what next?

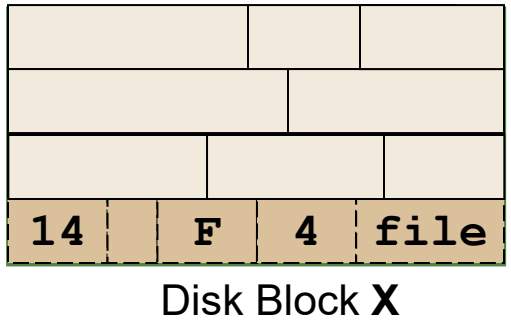



Listing **/.../.../BigBang**



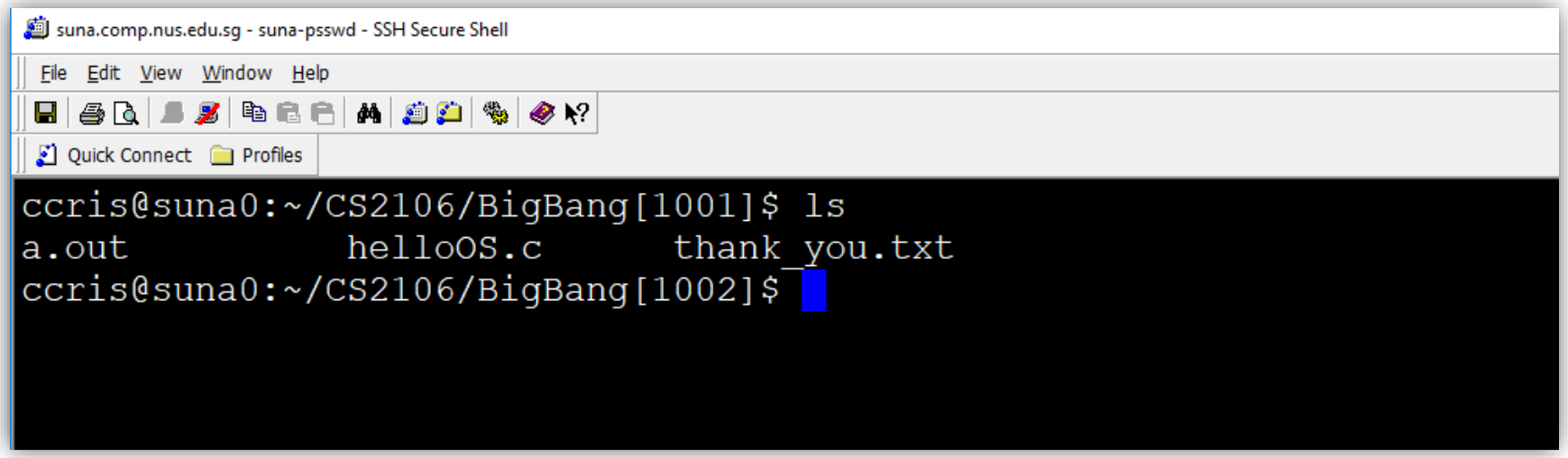

# Listing /.../.../BigBang



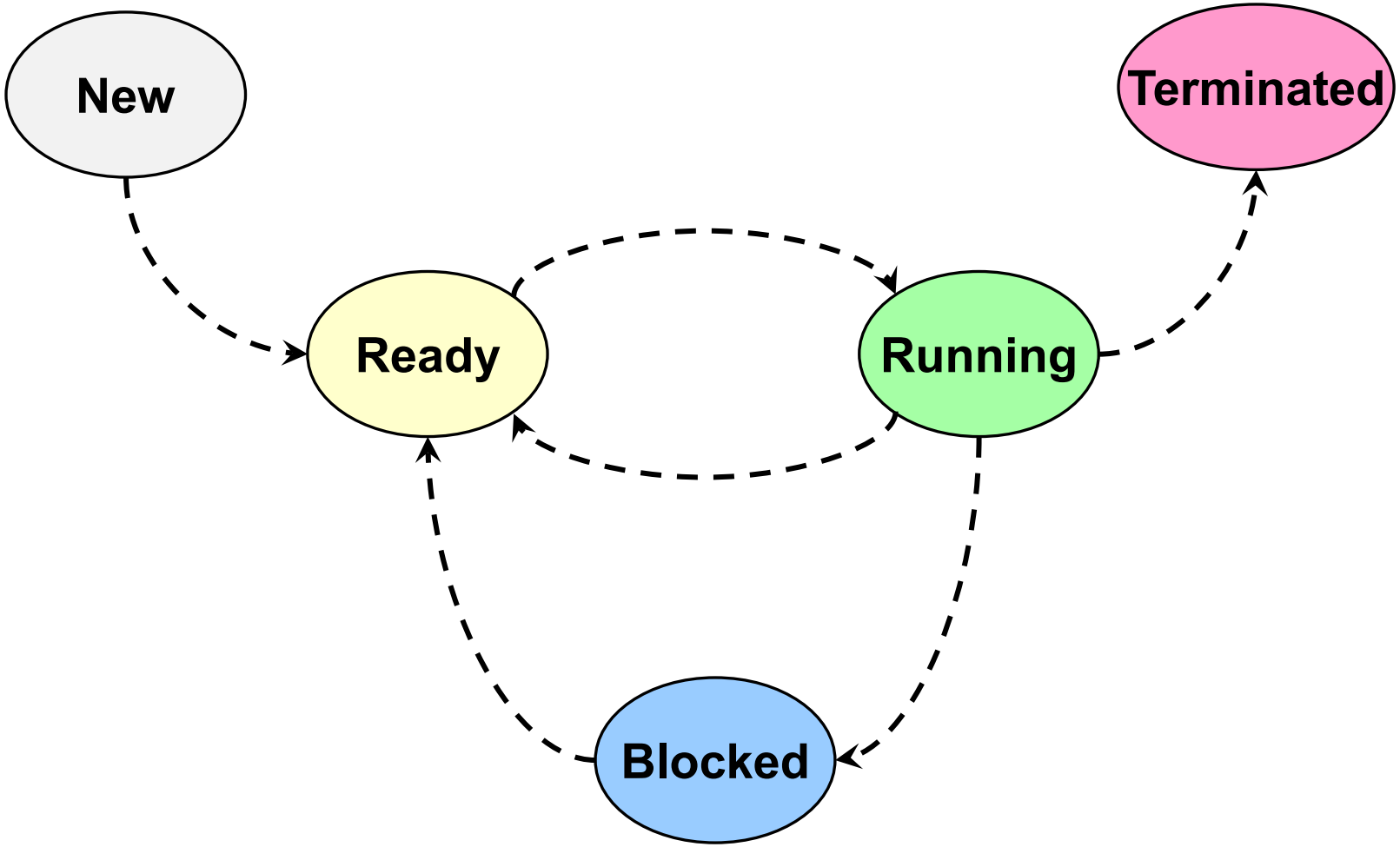




ls –l

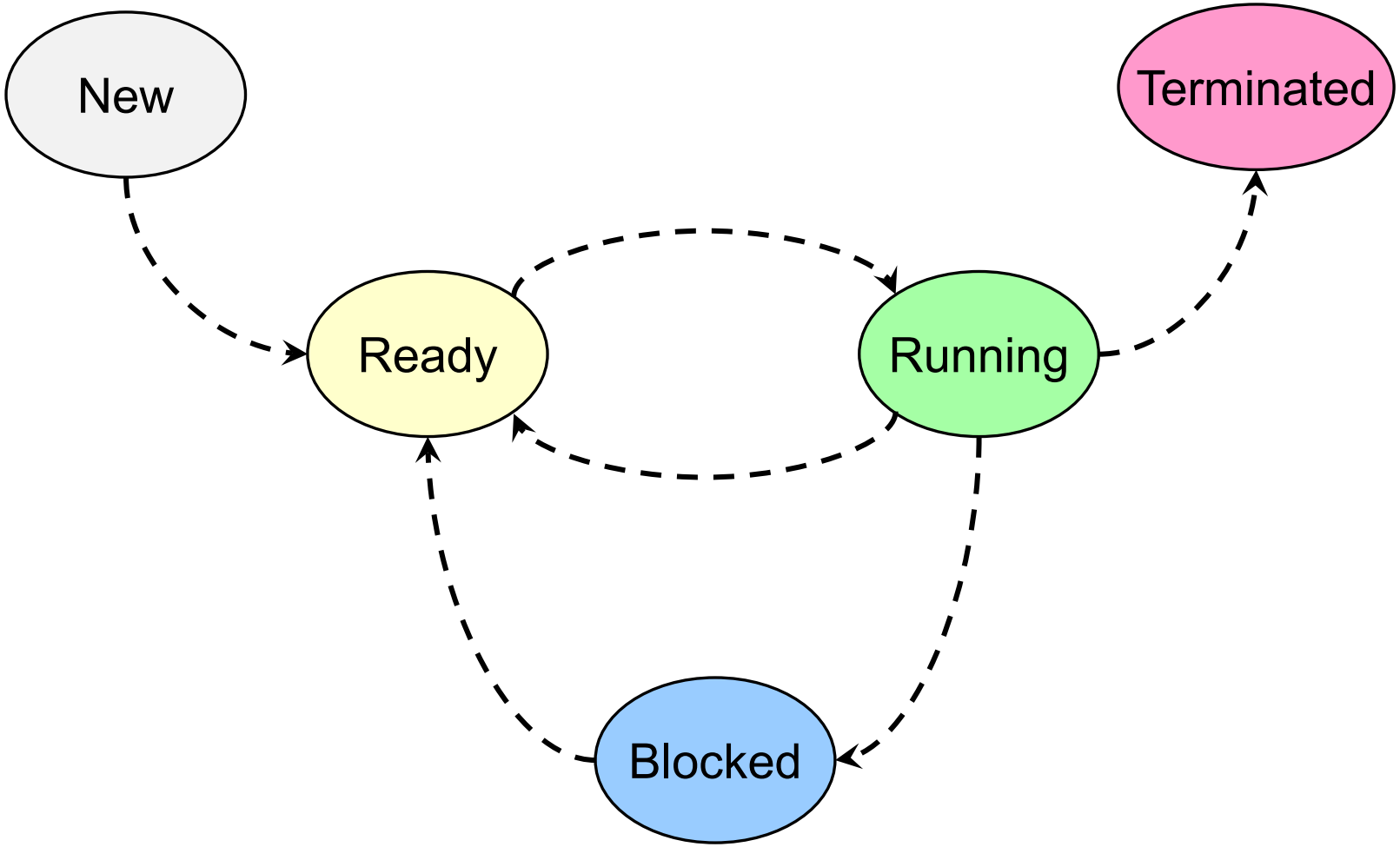
# "**ls**" prints the directory content



# Child **exits**



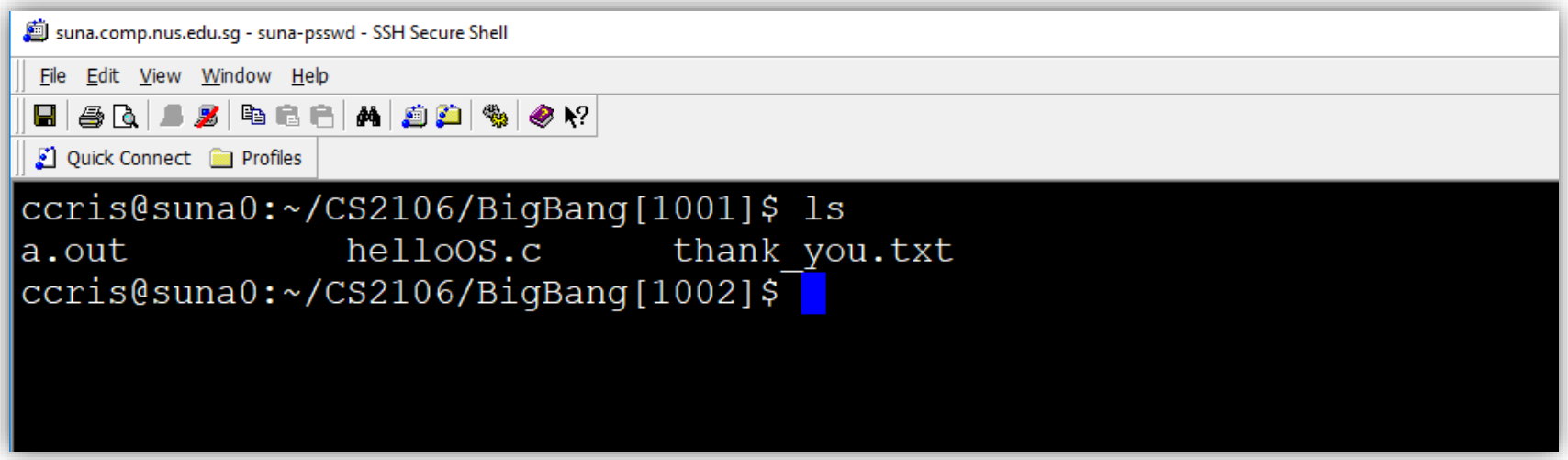
# The interpreter **hears about it**….



# Typical steps for Shell Interpreter

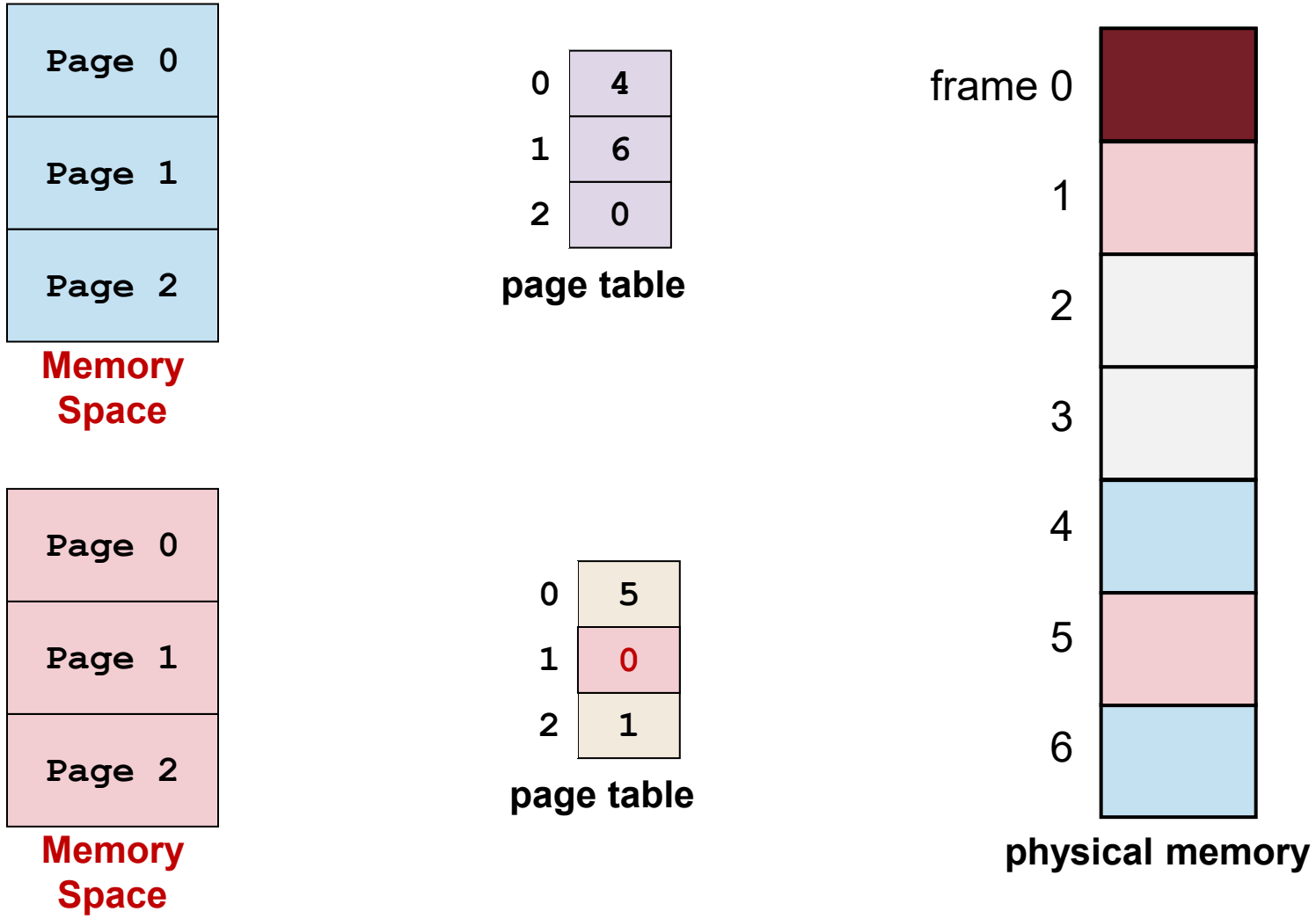
```
UserCmd  ← Read from keyboard
fork()
if I am the parent (i.e. the shell)
    wait ( child to finish )
else
    exec( UserCmd )
```

# Woohoo!!

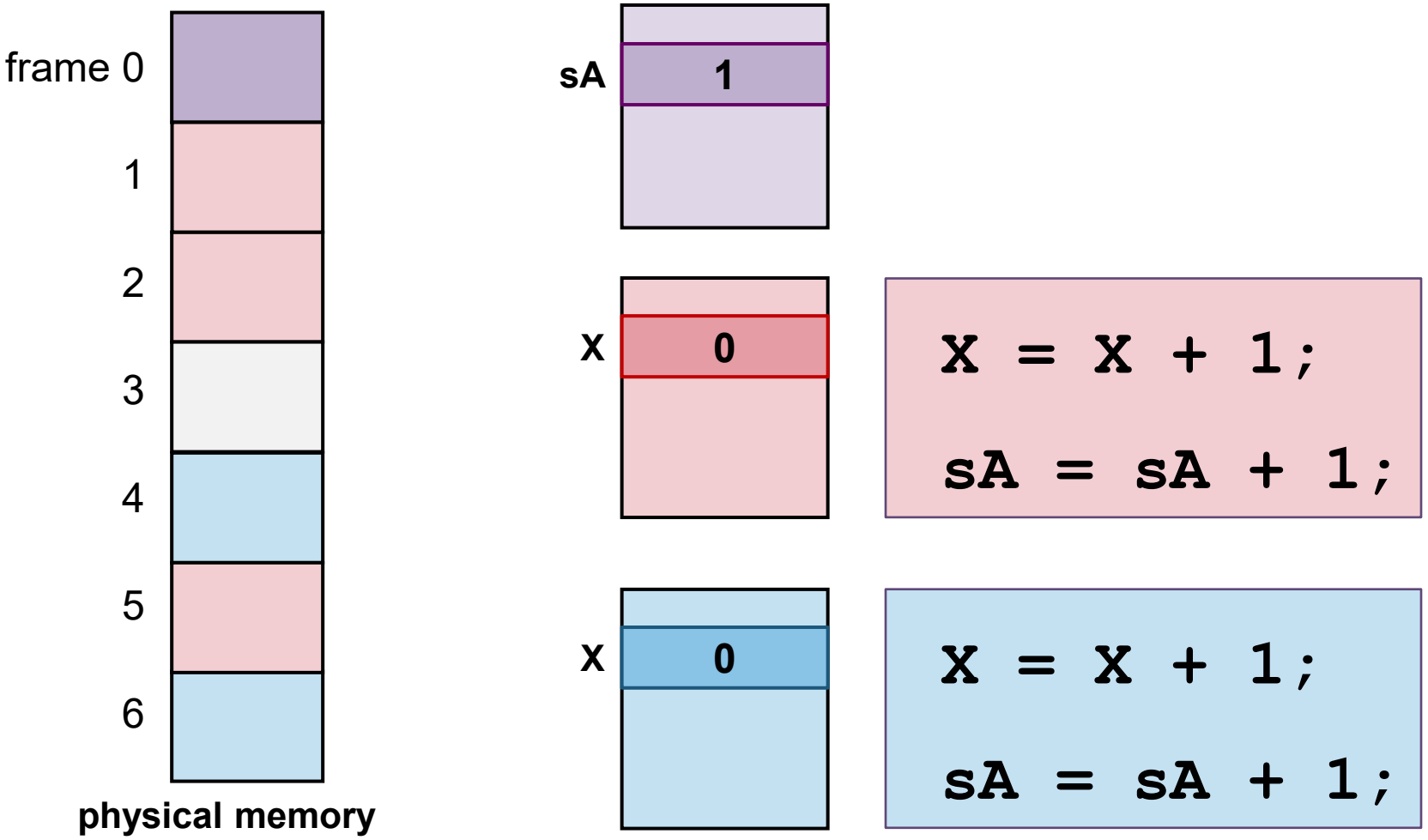

suna.comp.nus.edu.sg - suna-psswd - SSH Secure Shell
File Edit View Window Help
Quick Connect
Profiles
ccris@suna0:~/CS2106/BigBang[1001]$ ls
a.out helloOS.c thank_you.txt
ccris@suna0:~/CS2106/BigBang[1002]$

# WE SHOULD SHARE!

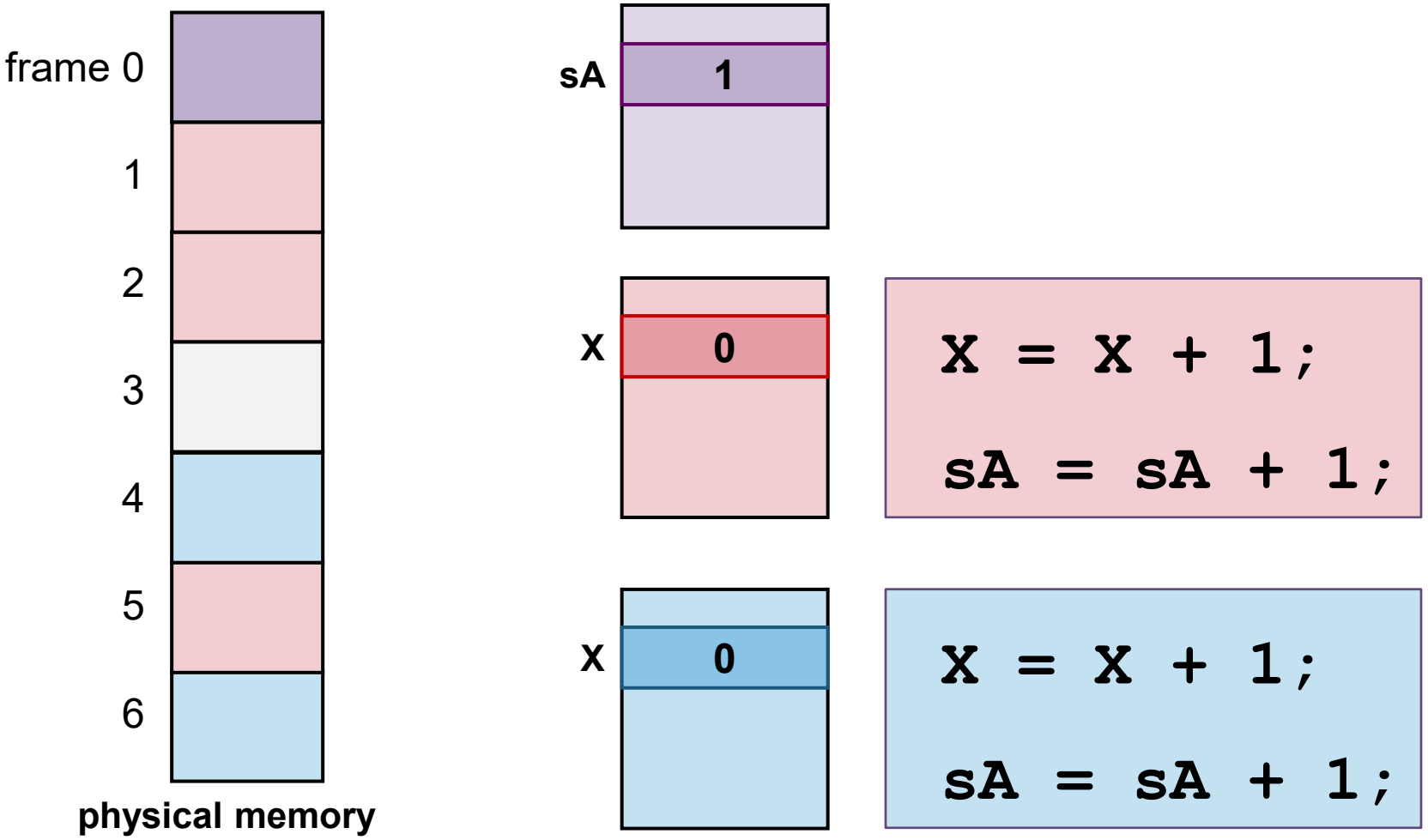
# How to **Share Memory**?



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



# Semaphore to the rescue!



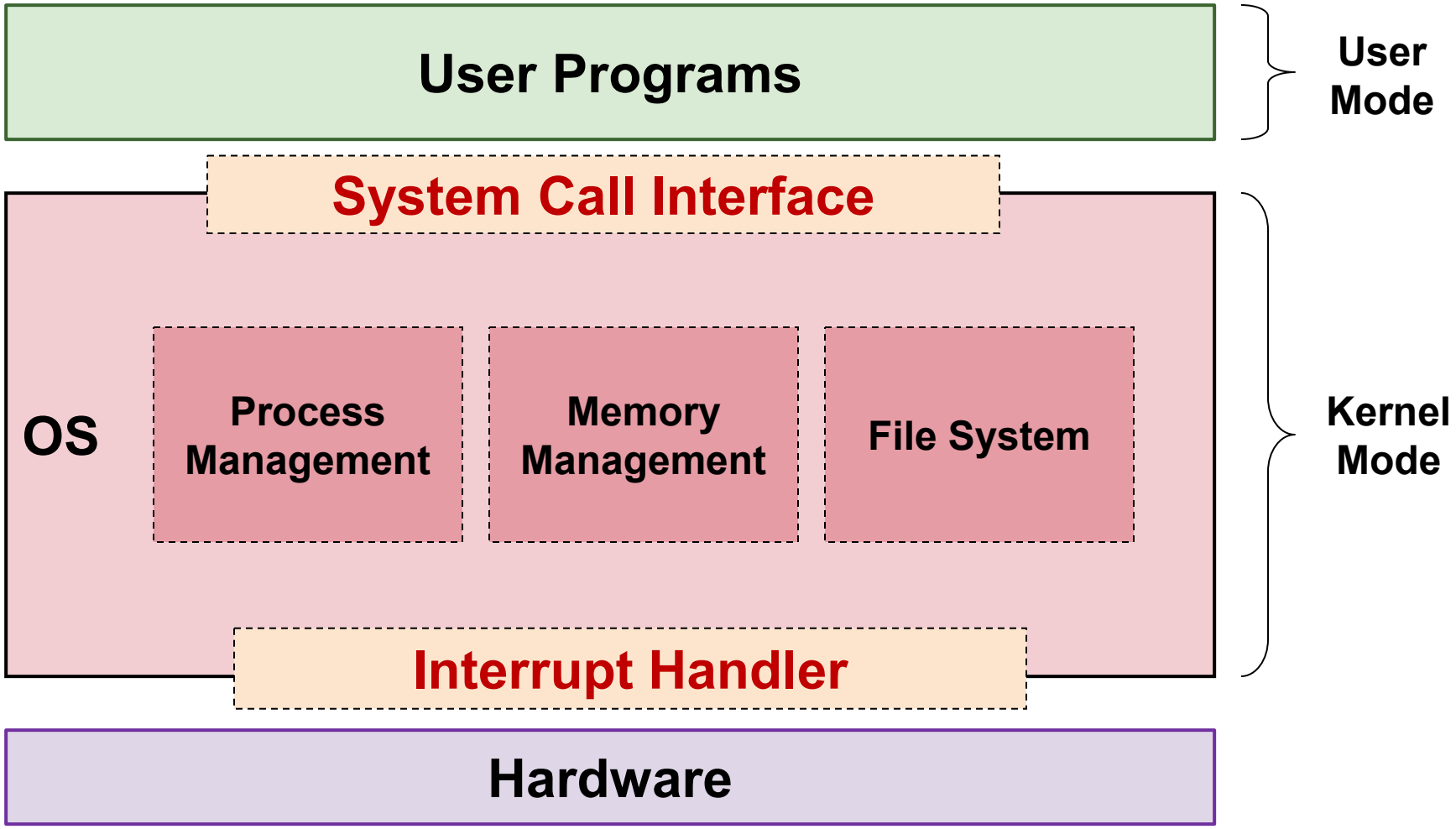
# Concurrency

- 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!