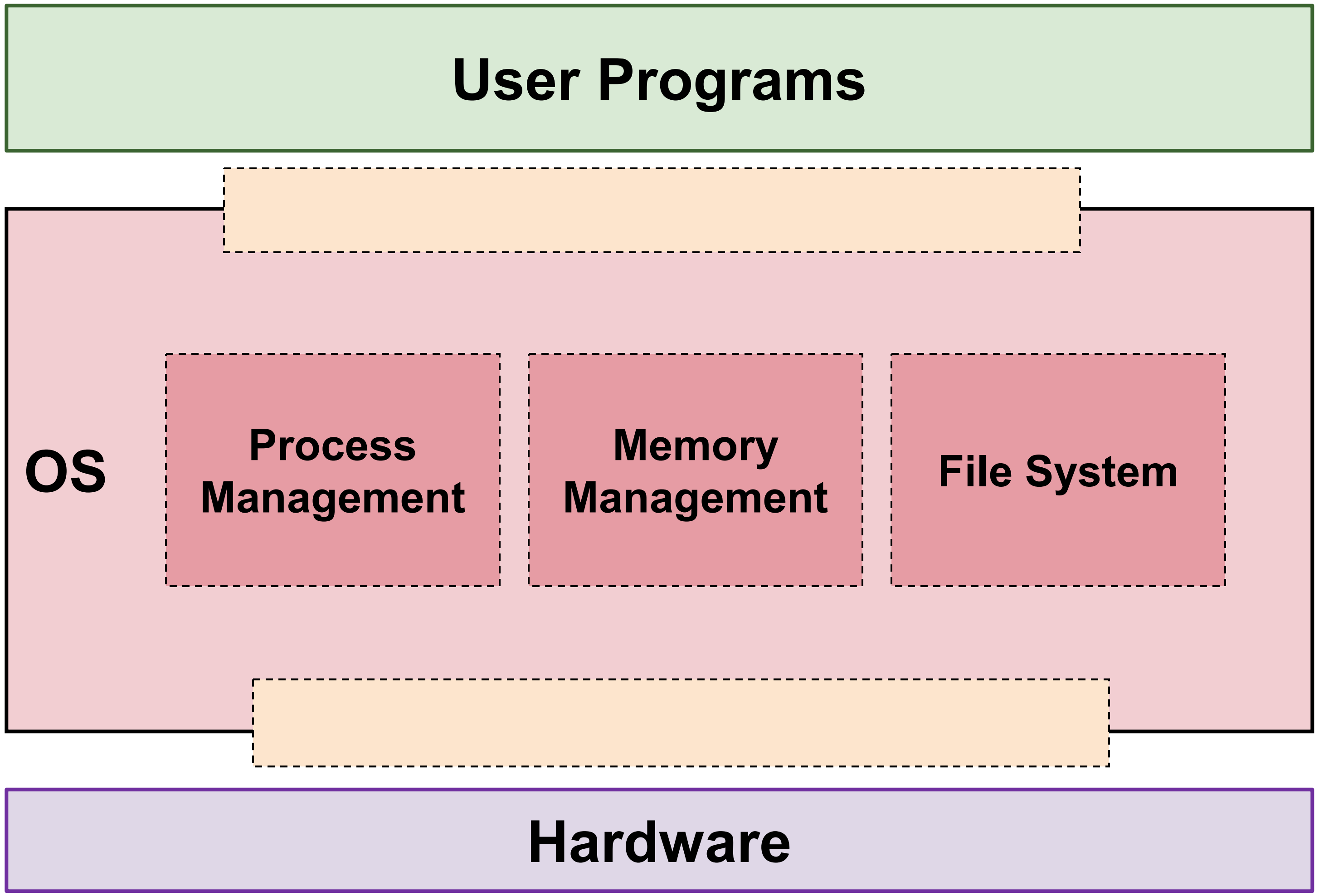
Big Bang & Revision

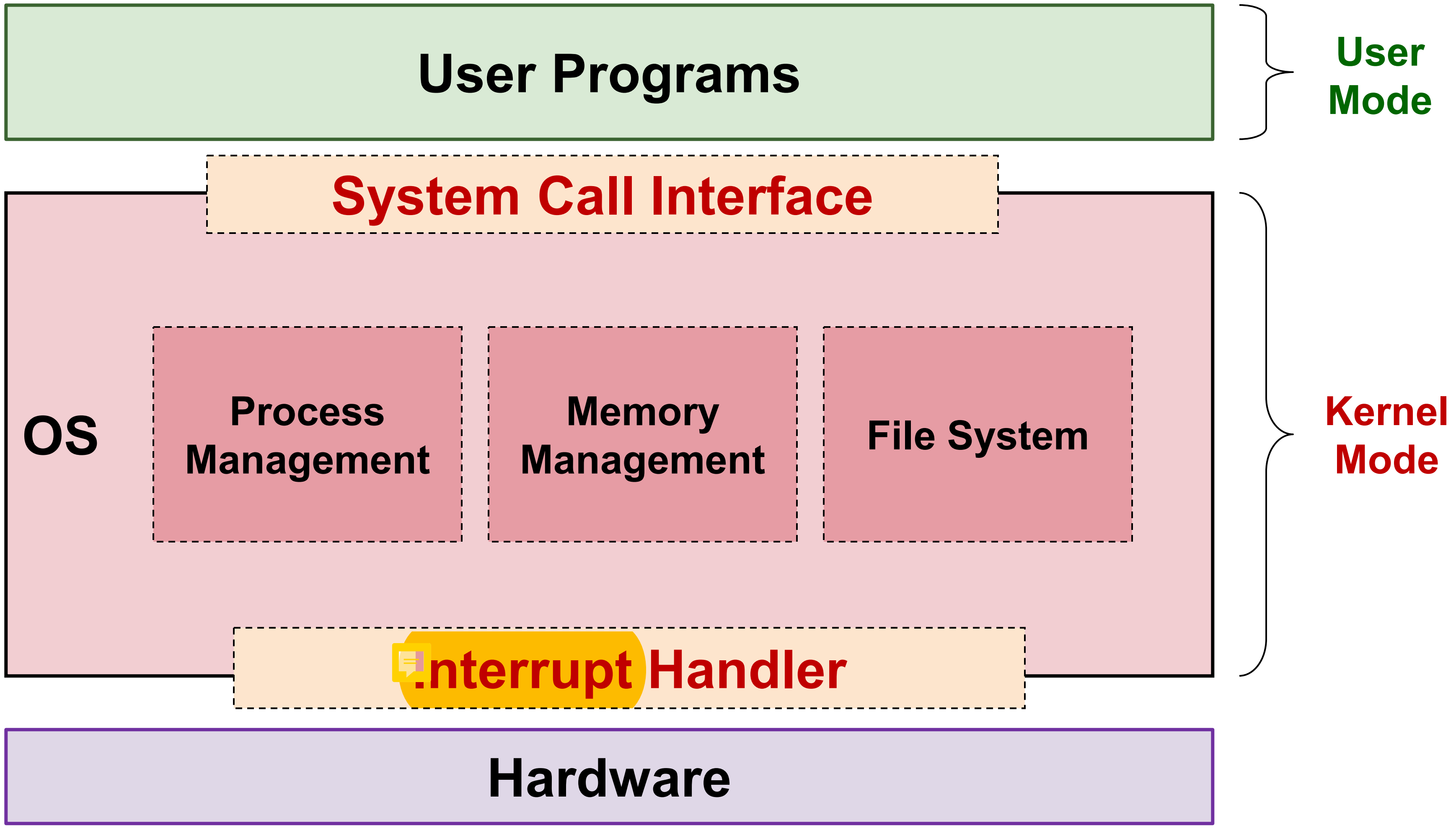
Lecture 12 (**Live Version**)

OS Structure (Monolithic)



OS Structure (Monolithic)

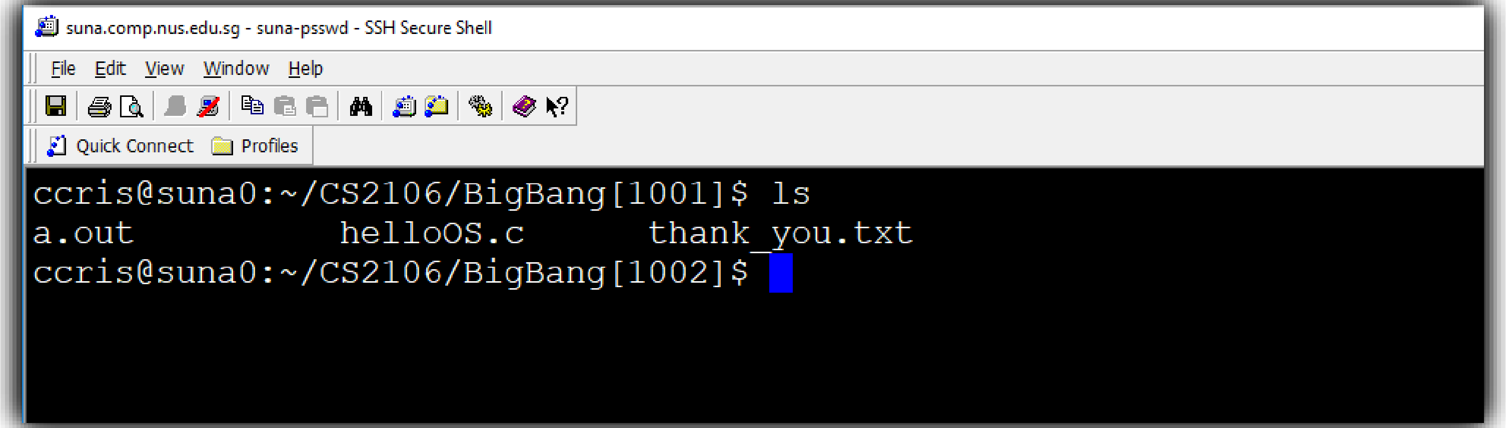
exception is generated by the instructions which are run by the user or OS
ie: exception happen due to the code ran by user / OS



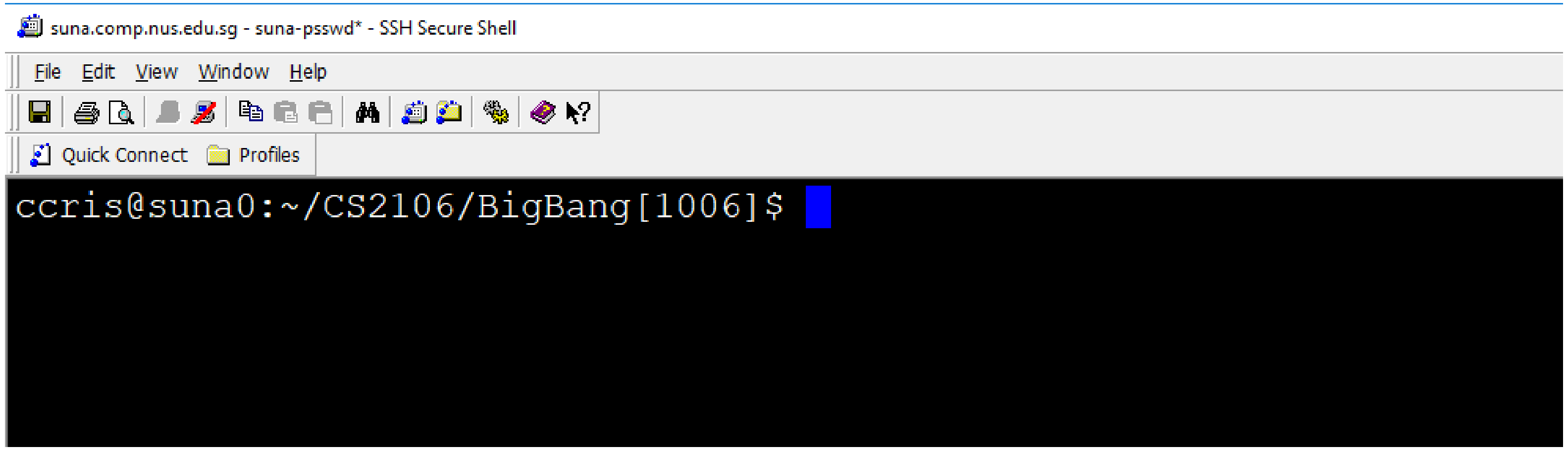
*23 + 11 + **lots of** hours just to learn about what happened in **1 second***

THE STORY OF
SHELL INTERPRETER

Ah…. Ah…. so simple….. **or is it**?



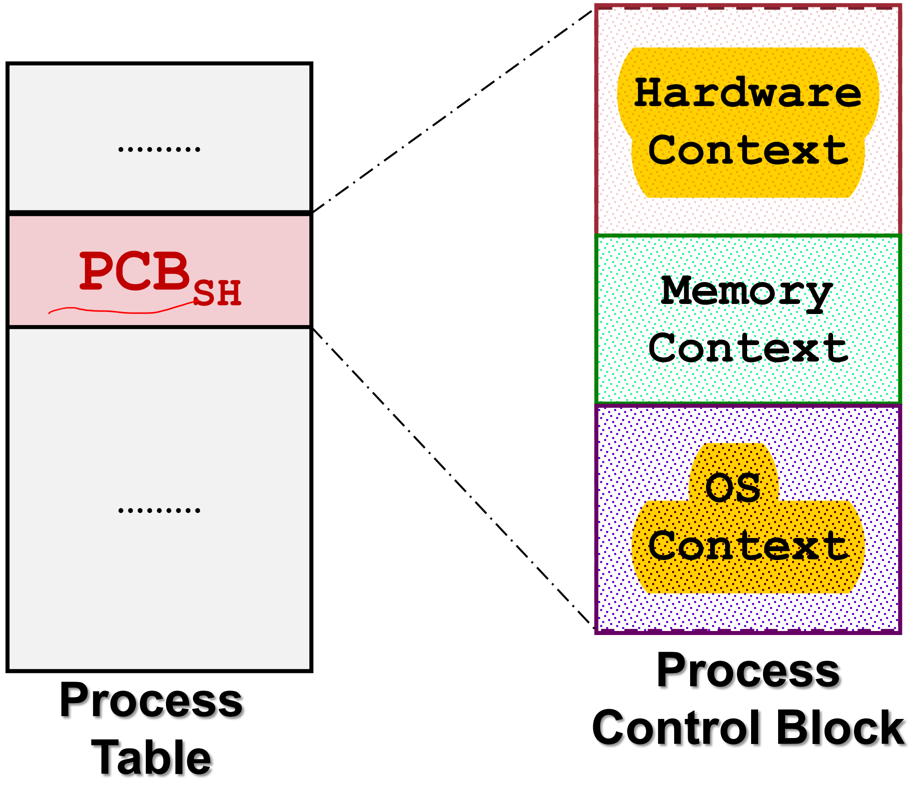
Here we go … …



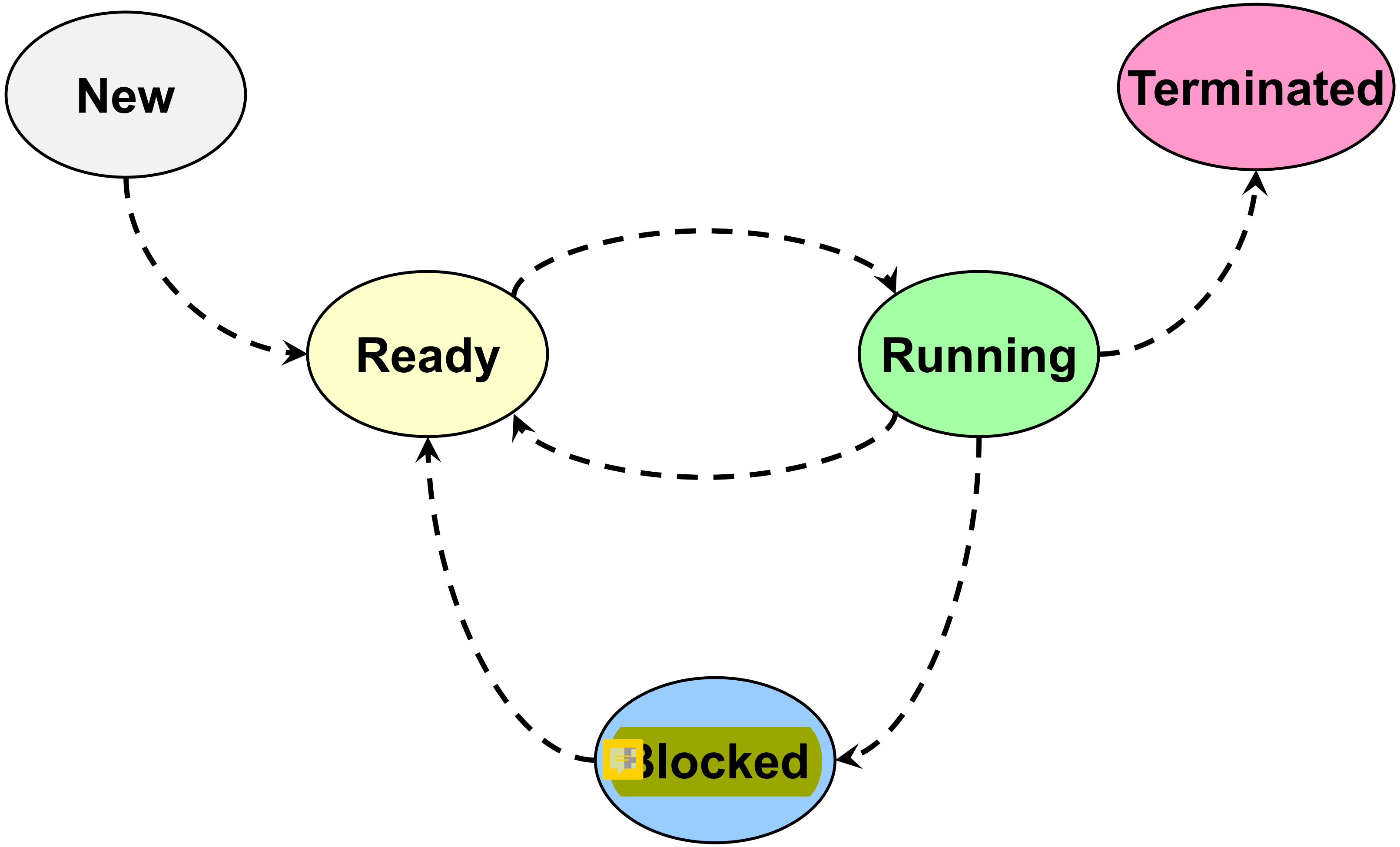
suna.comp.nus.edu.sg - suna-psswd* - SSH Secure Shell
File Edit View Window Help
Quick Connect
Profiles
ccris@suna0:~/CS2106/BigBang[1006]$

Process Table & Process Control Block

the shell interpreter is waiting for input - and since it is a process it will have an entry in the process table and its own process control block



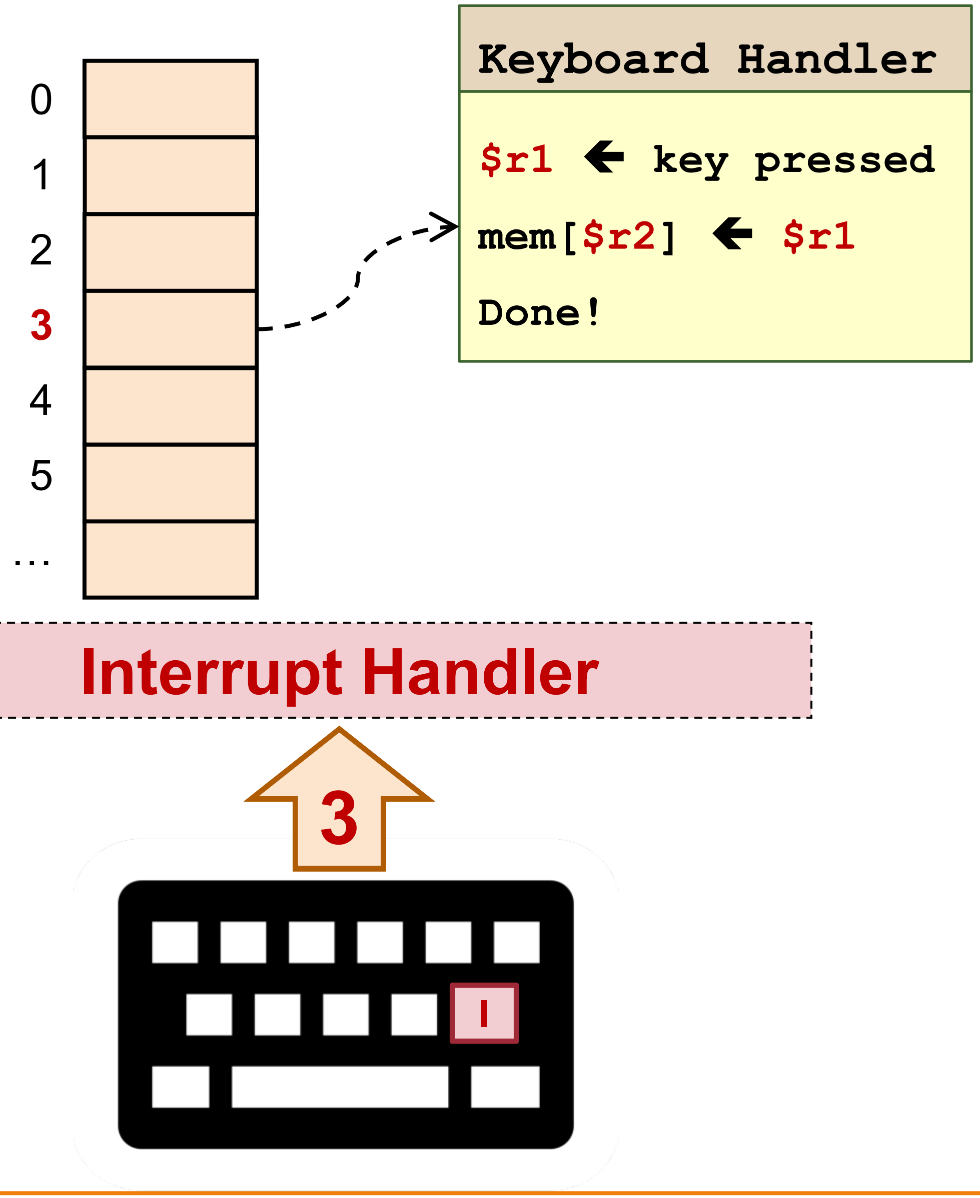
The Interpreter is in…..

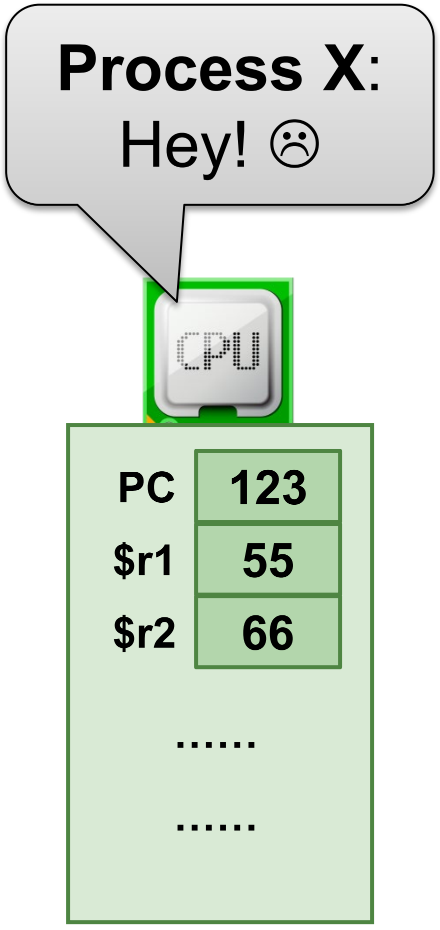


User press "1"

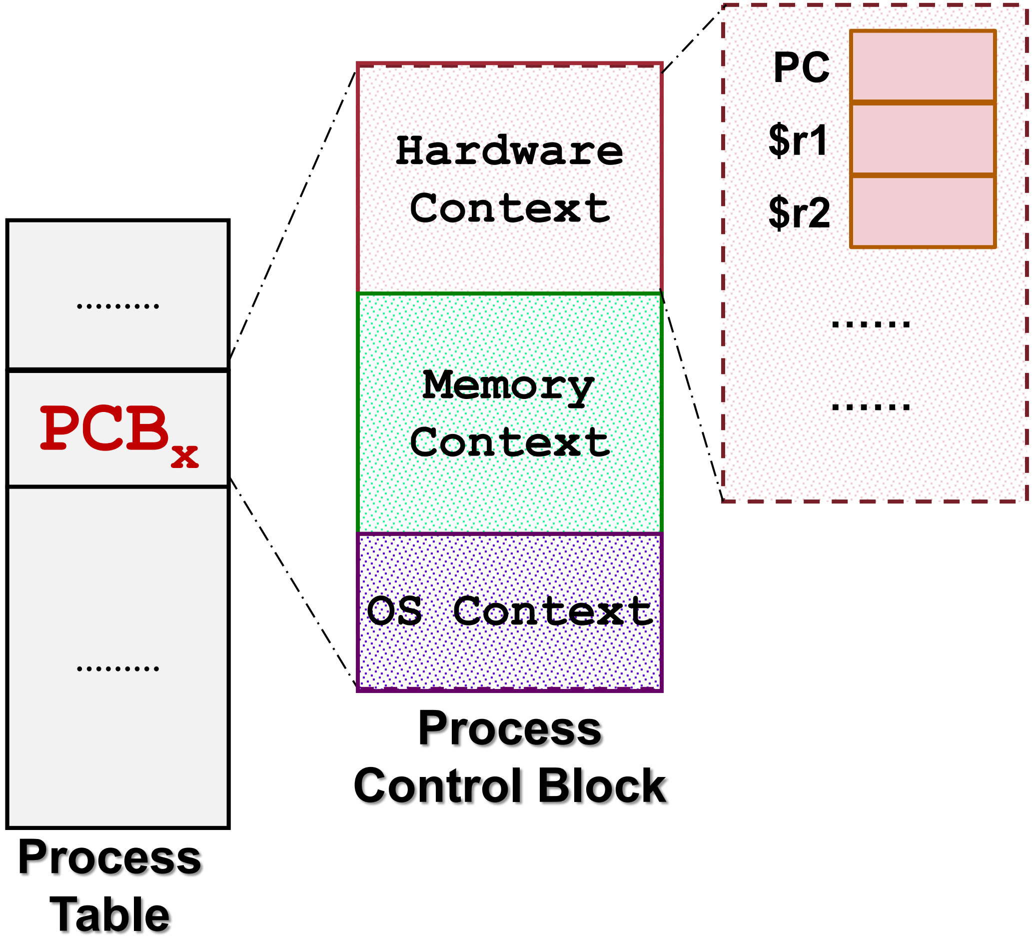
if (iRQ == 3) then
call Keyboard Handler();

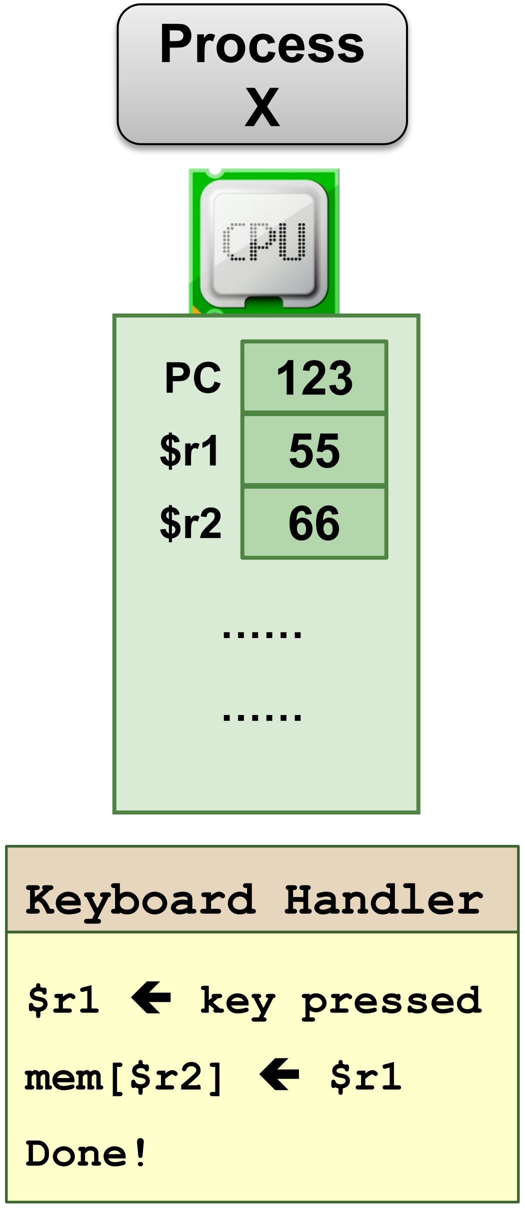
or in this case can use function pointers to call the handler





Sorry to interrupt you….

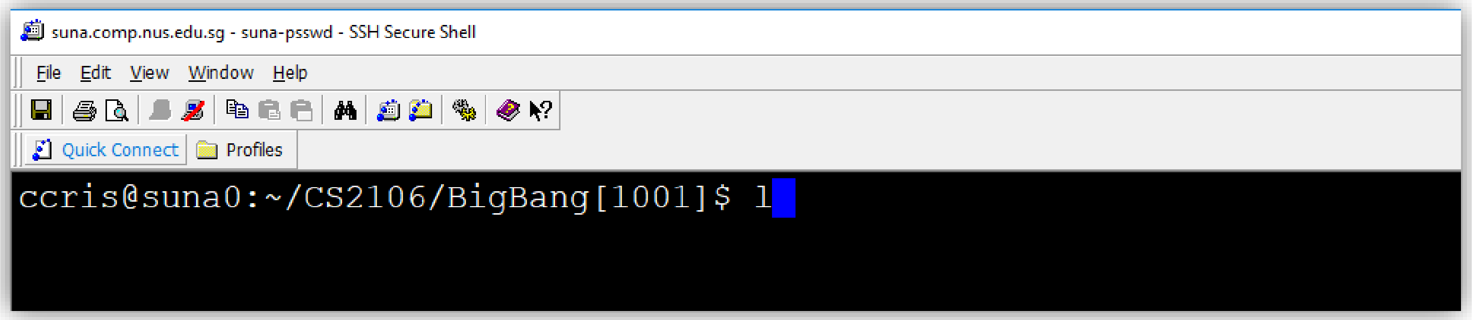


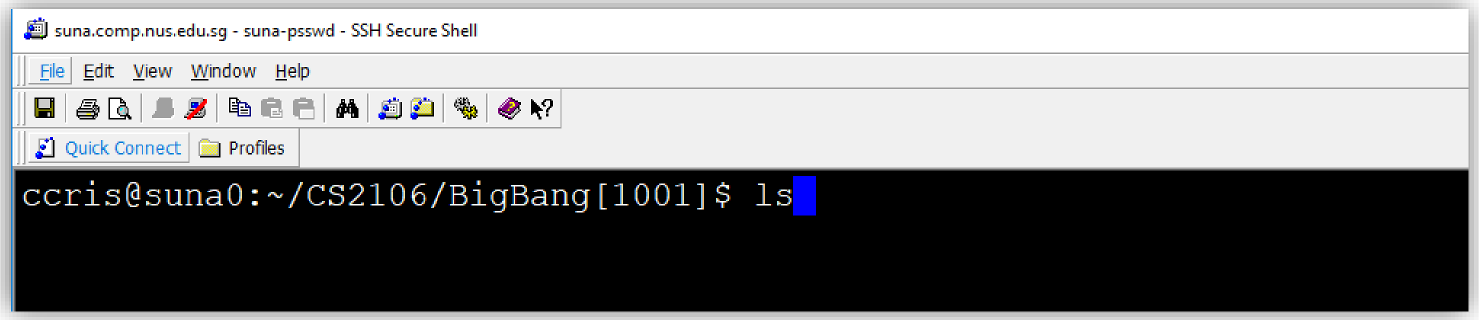


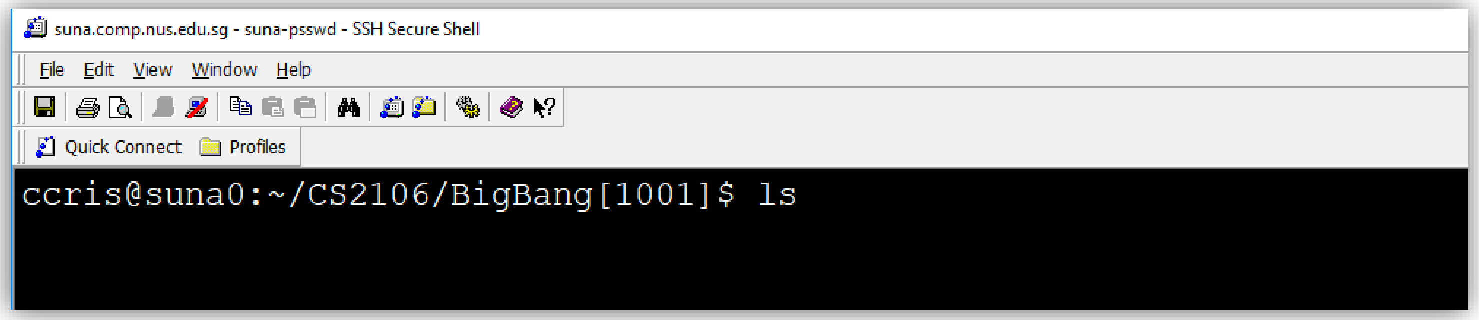
Interrupt steps

- 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

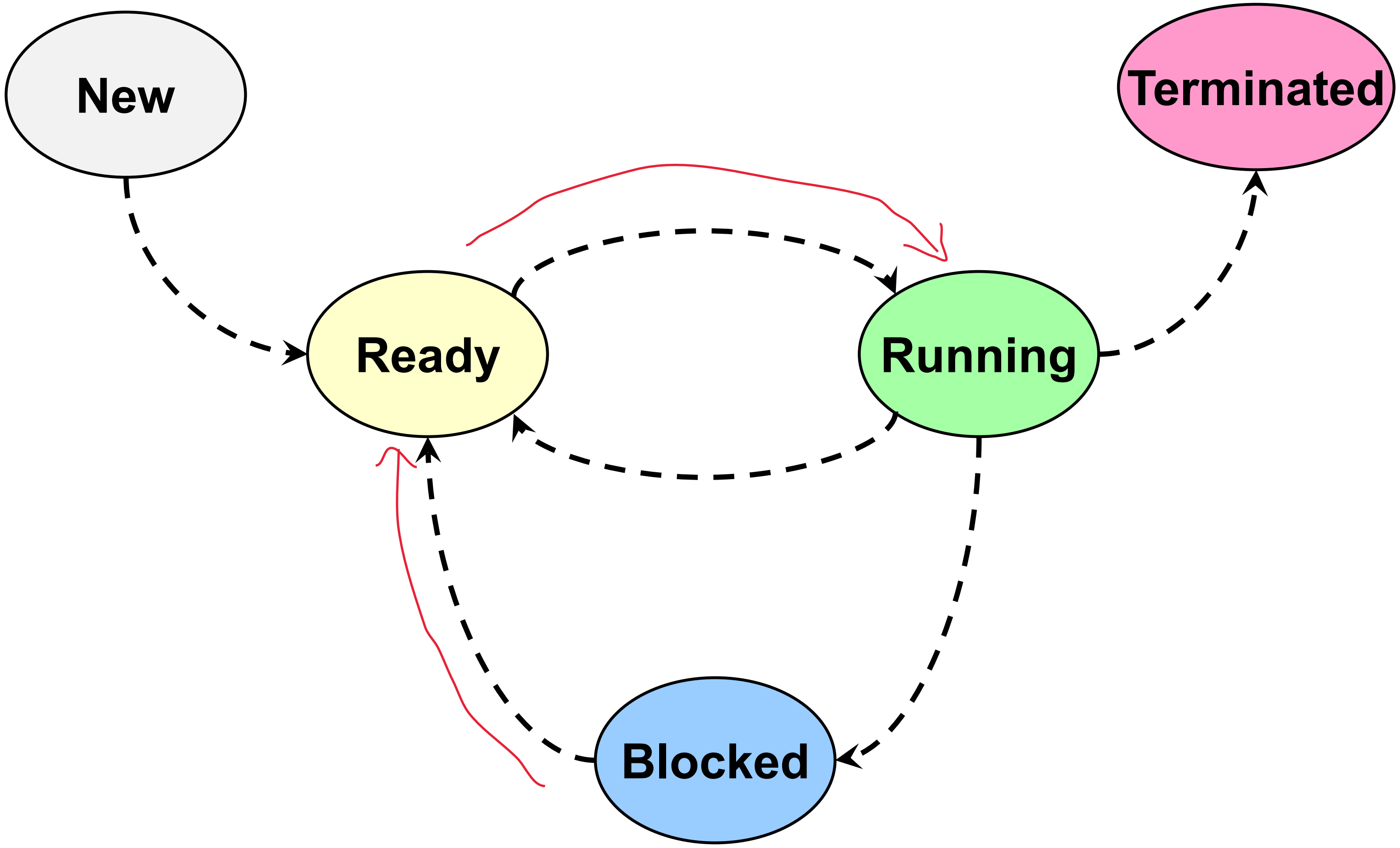
Rinse and Repeat……



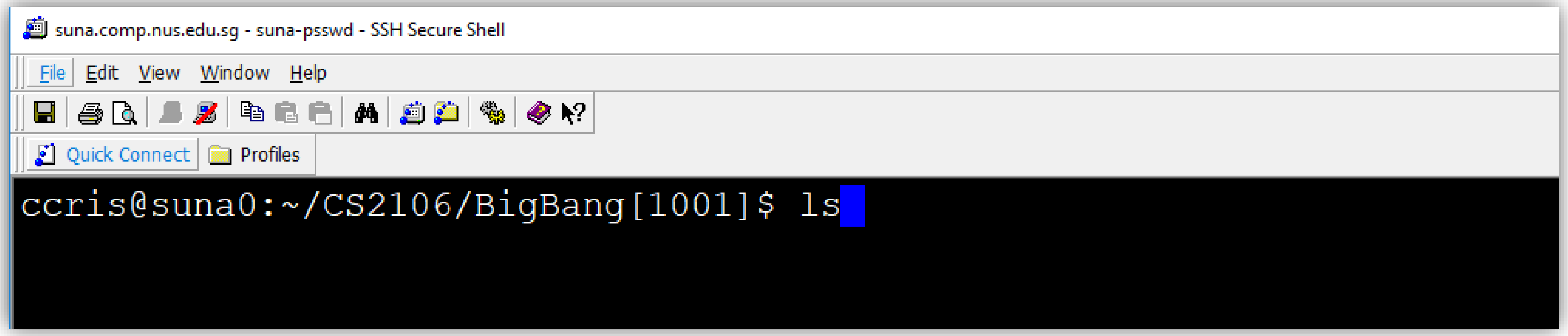




The **interpreter** is now…



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

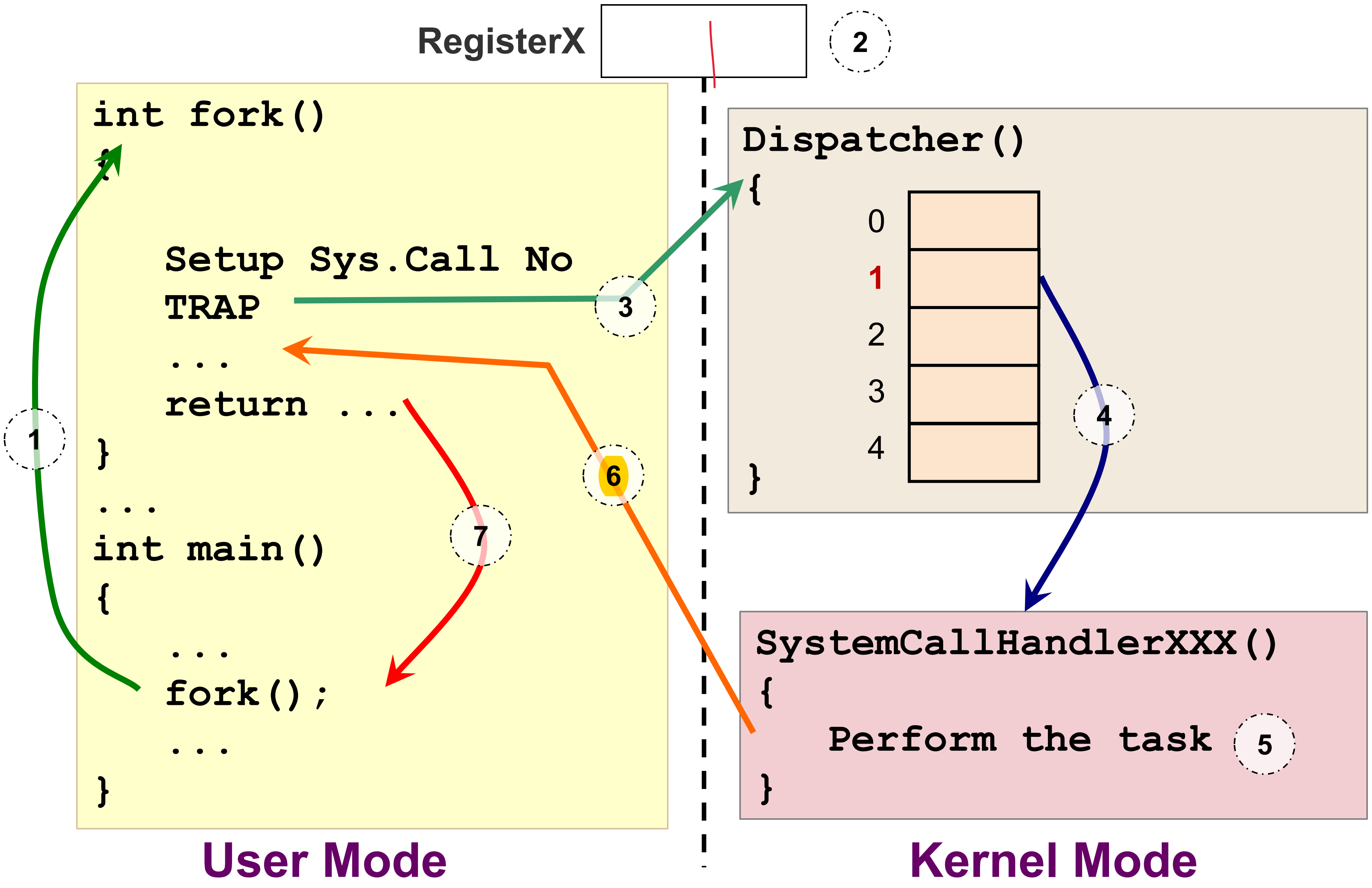
```
int fork()
{


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

```
Dispatcher()
{


}
```

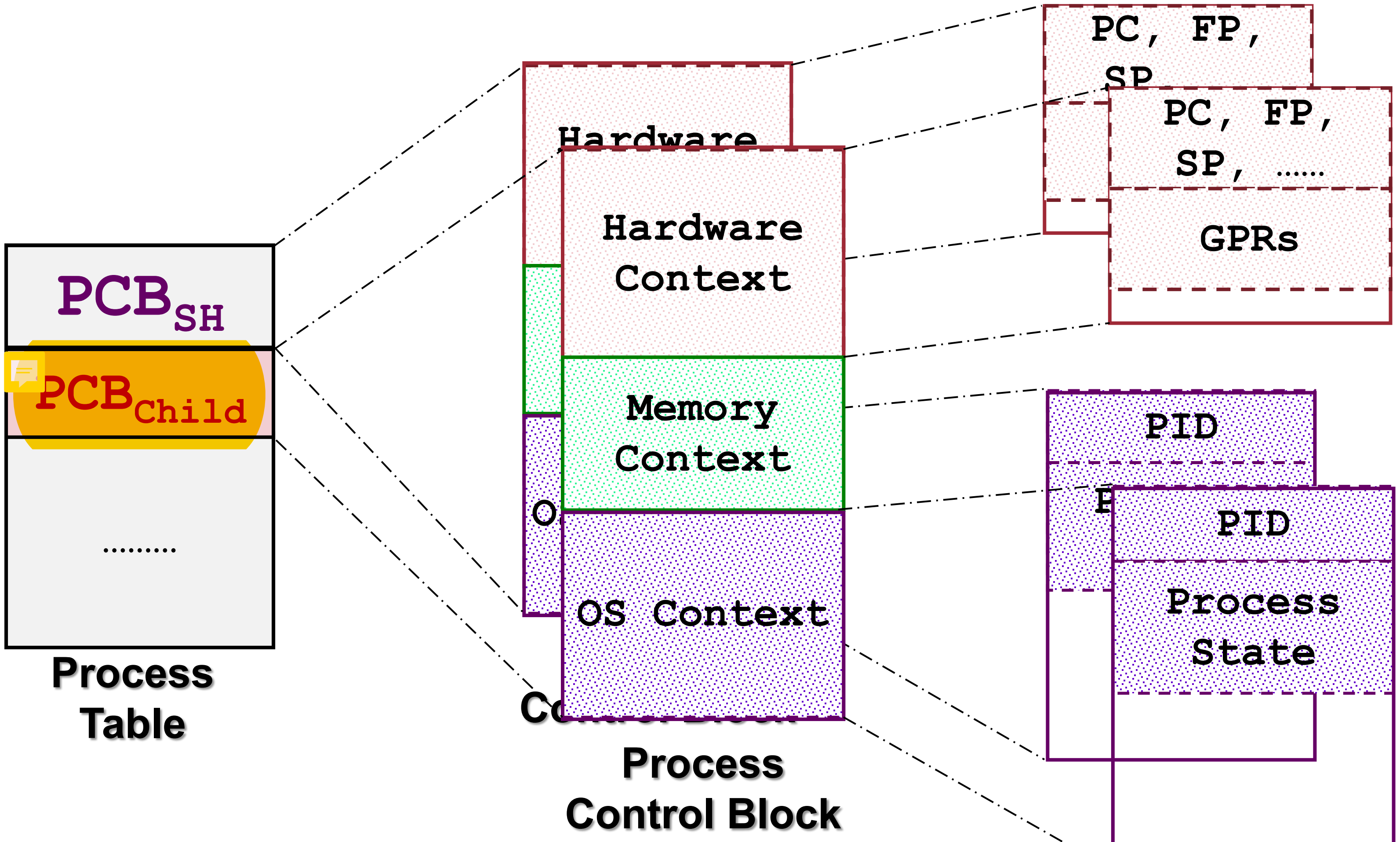
```
SystemCallHandlerXXX()
{

}
```

User Mode | **Kernel Mode**

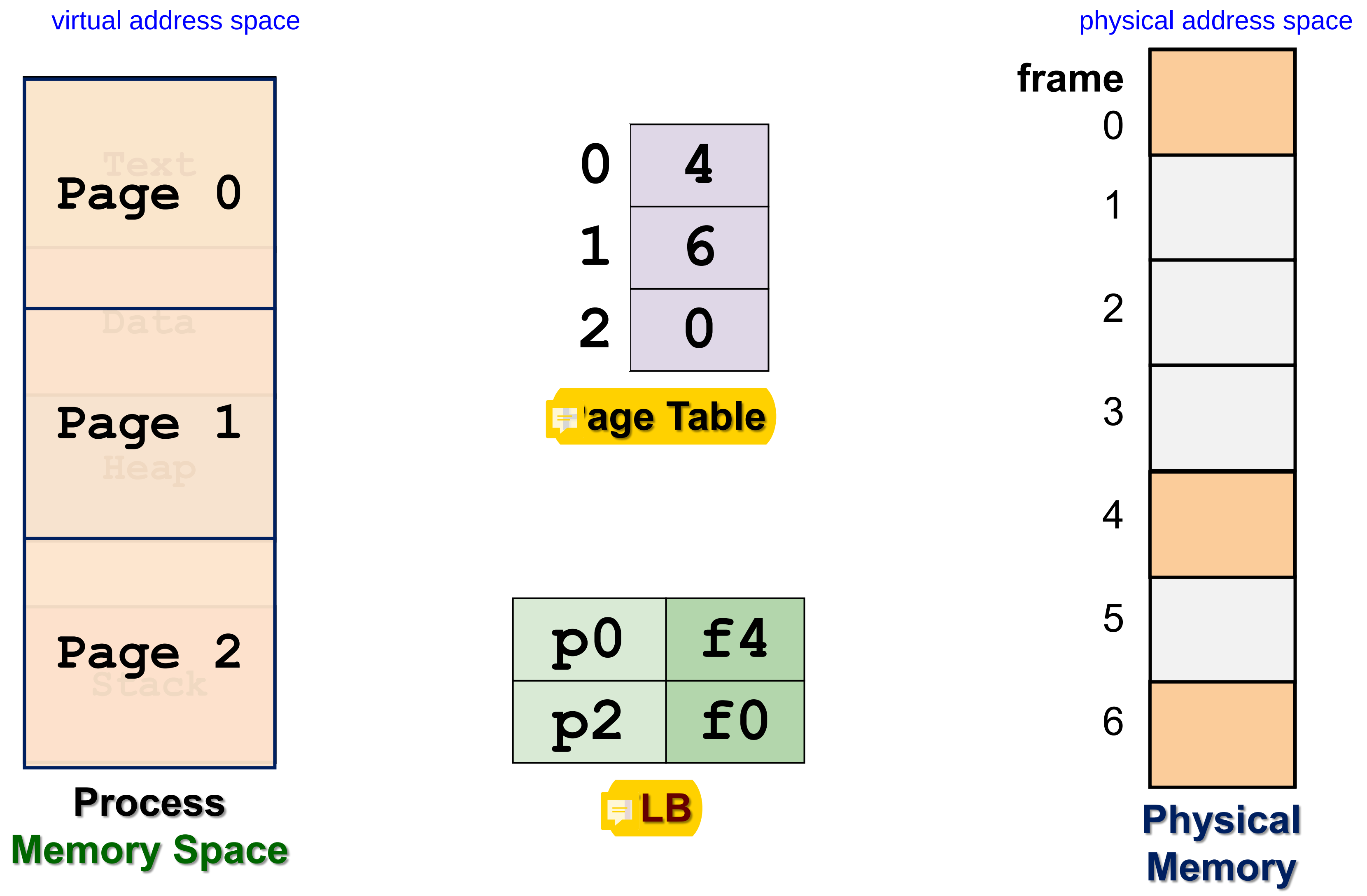
fork() involves a system call



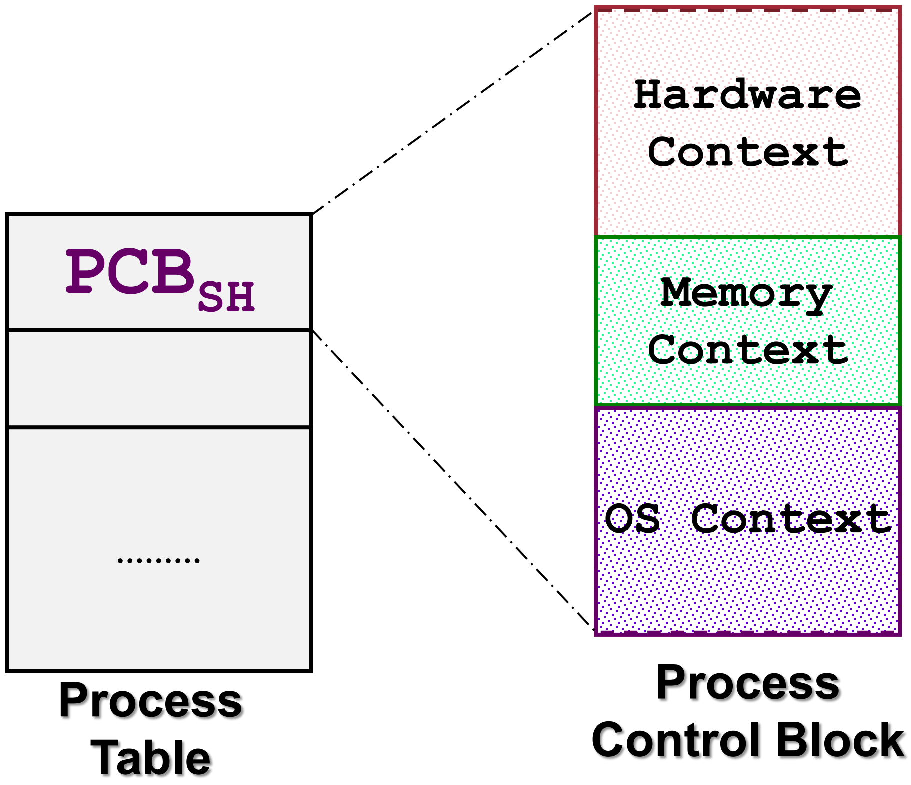
What is the **effect** of fork( )?



Memory Space of a Process

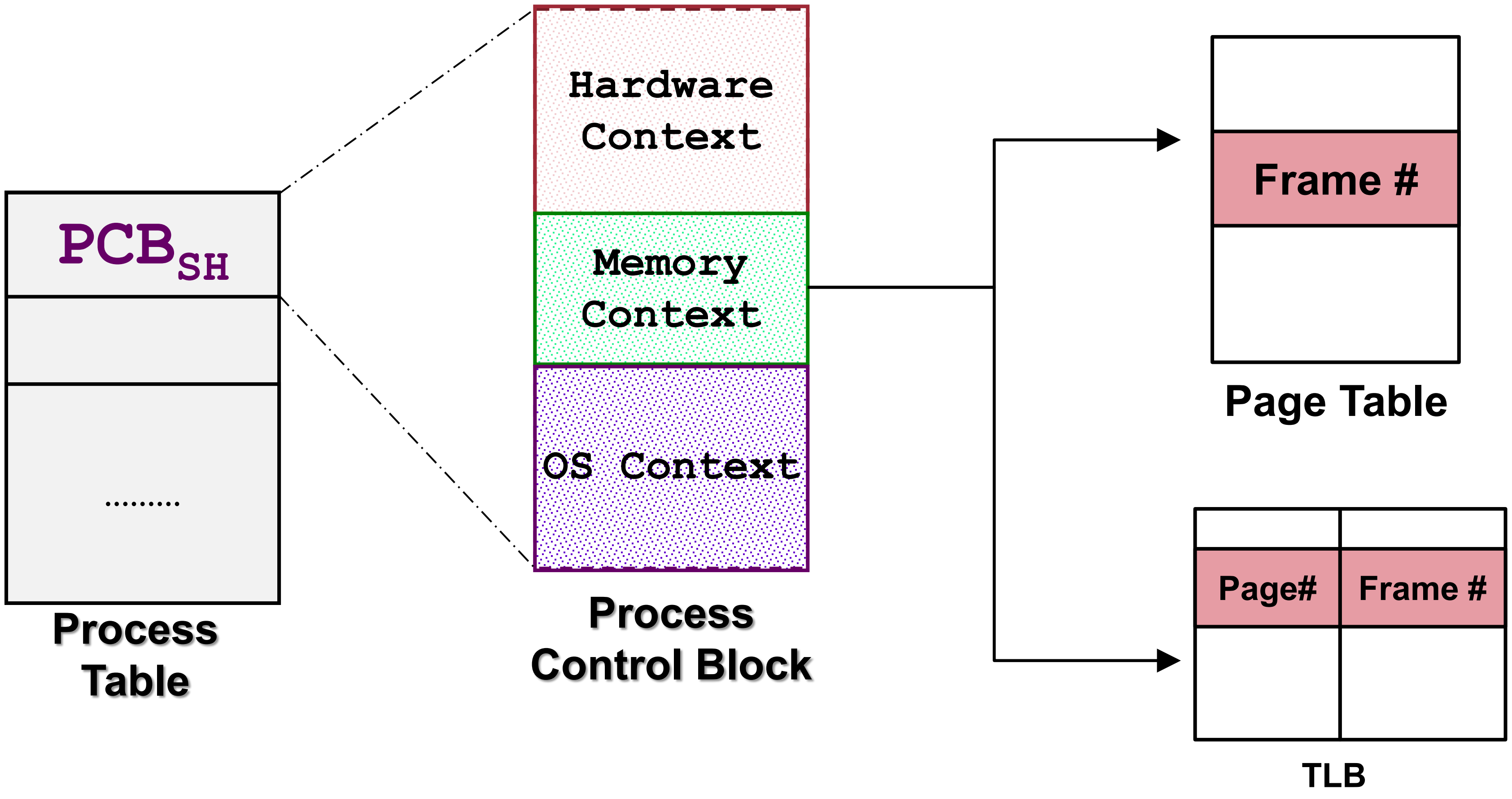


Memory Context = ?

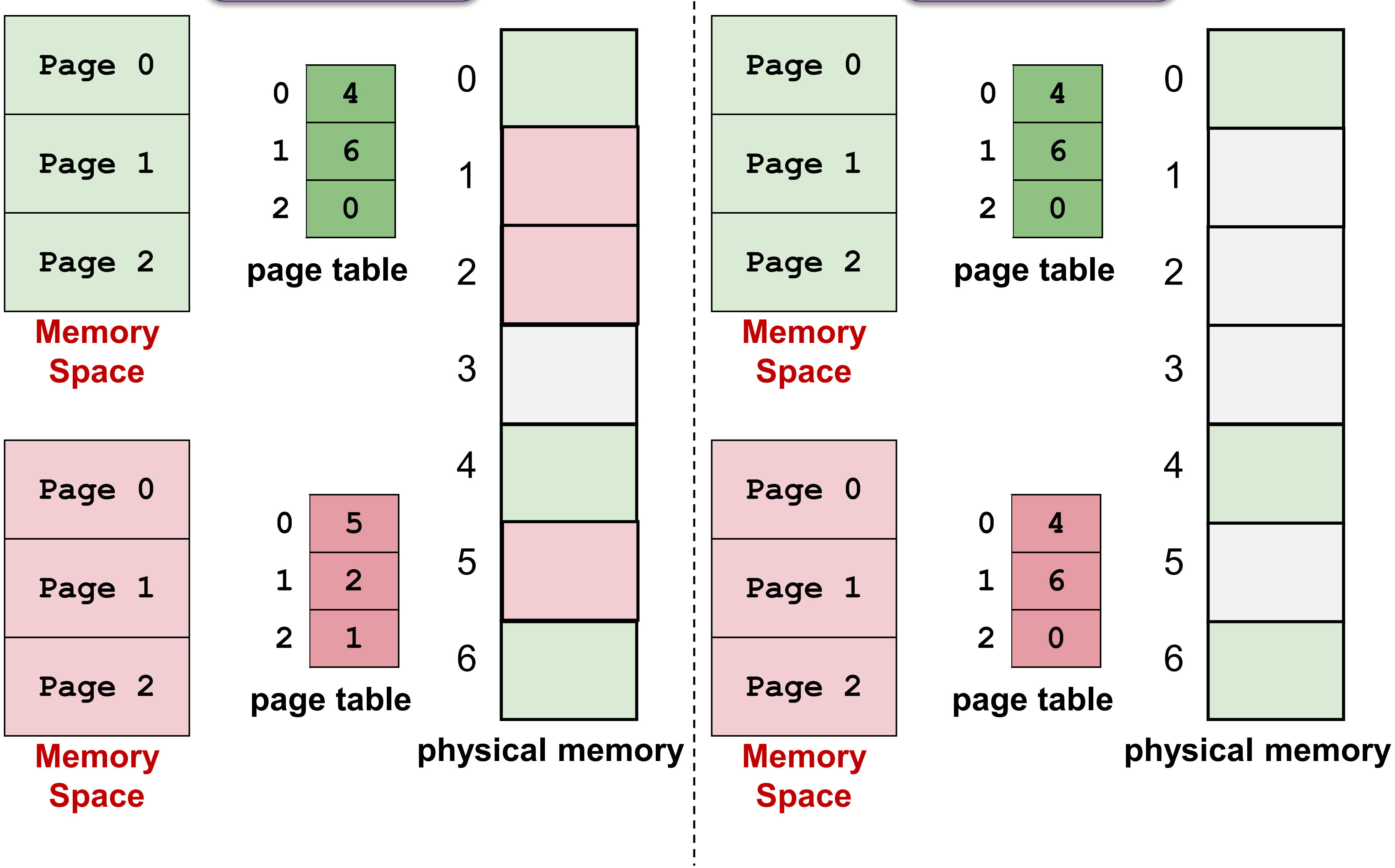


Memory Context = ?

memory context will store a pointer to the Page Table (must) and TLB (not all the time)

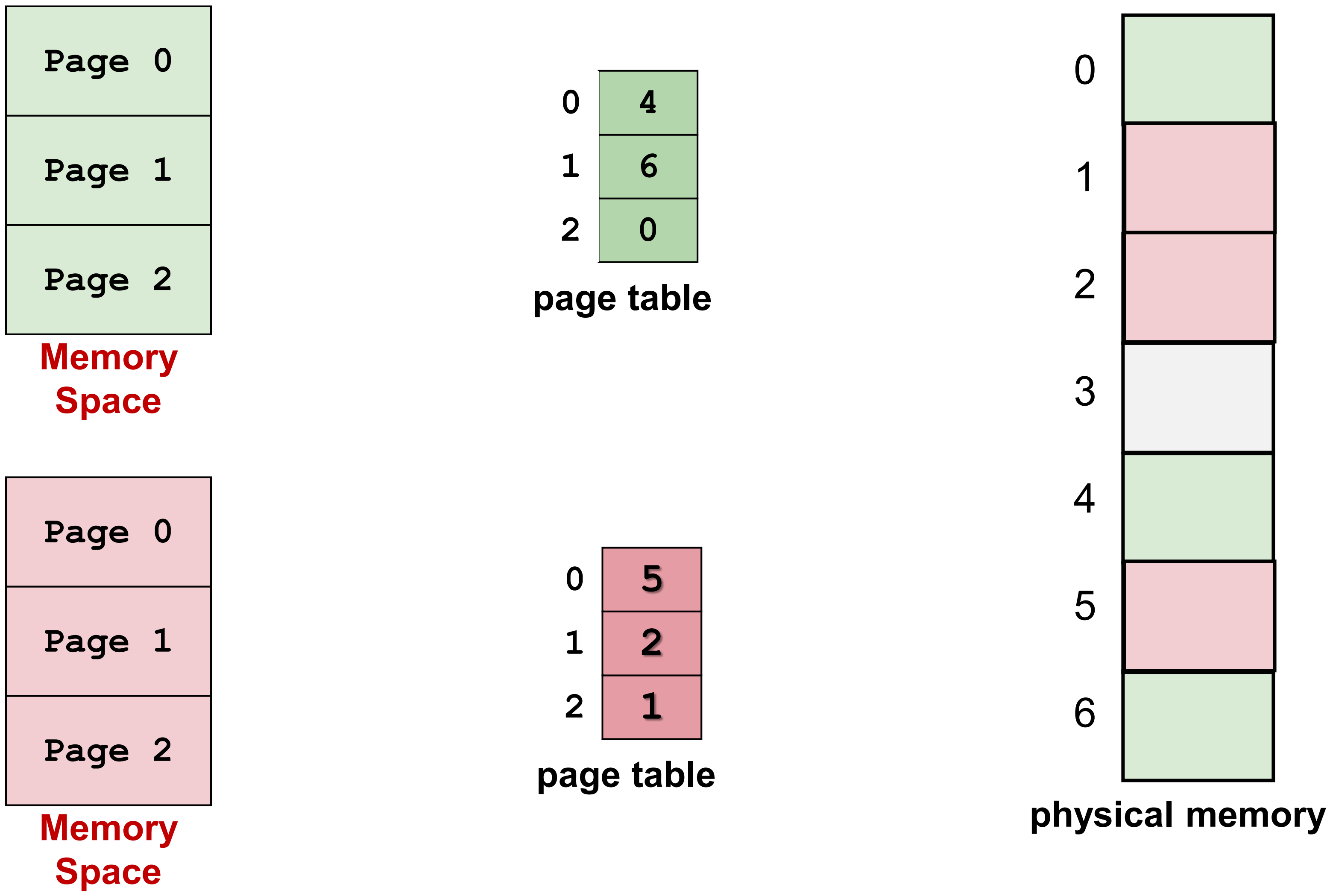


What is the b A of "duplicatio B

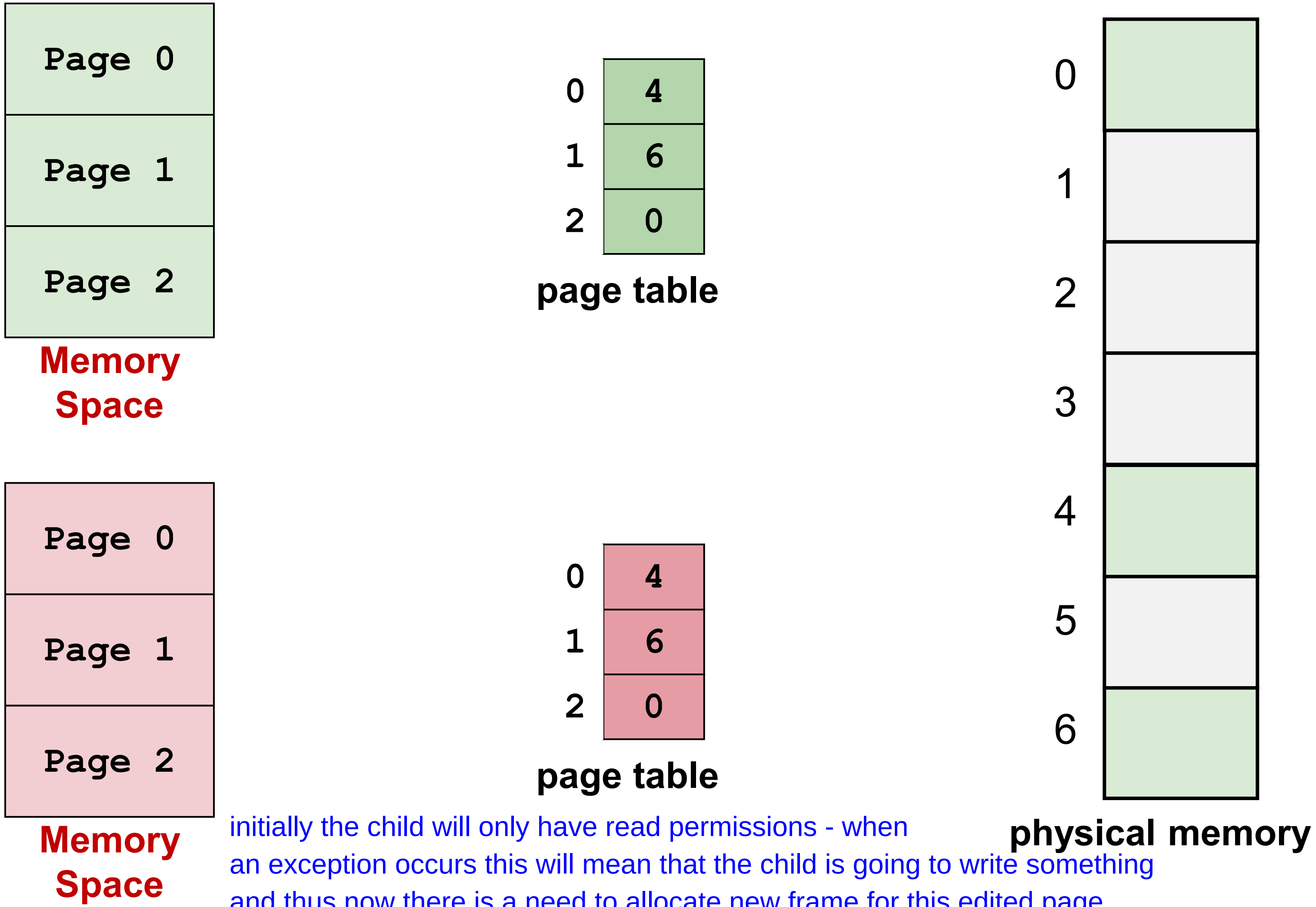


Duplicating Memory Space the HARD WAY

no need to do this since most of the time child process does not require to change much of the pre existing data

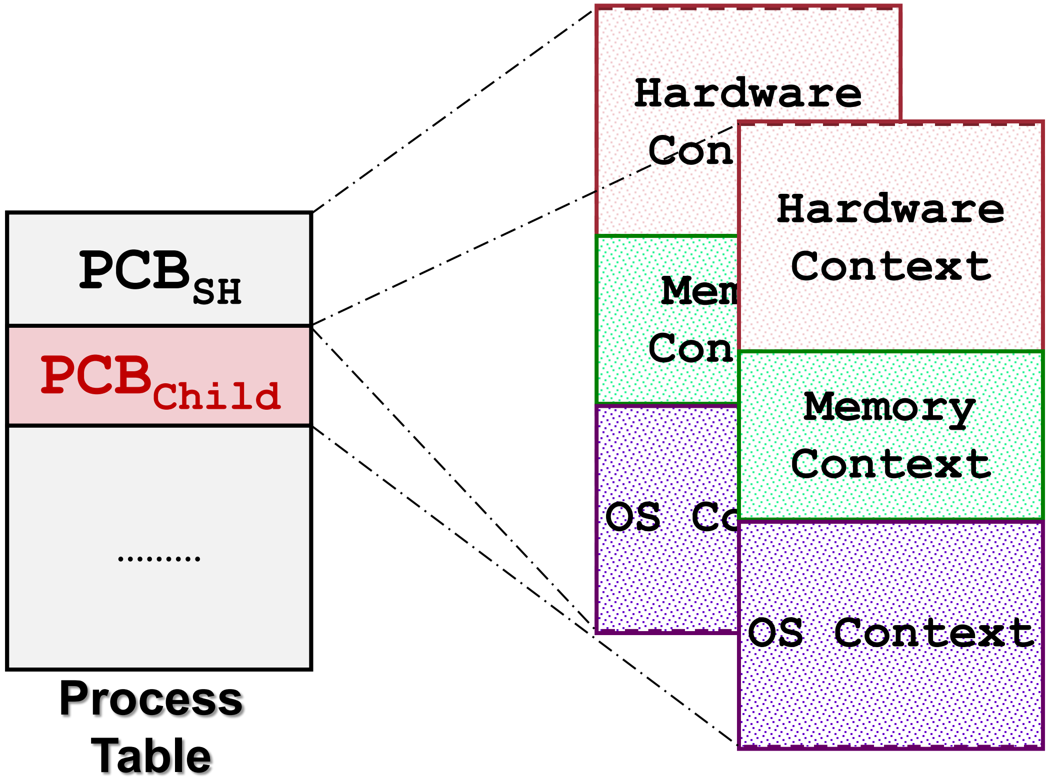


Copy on Write

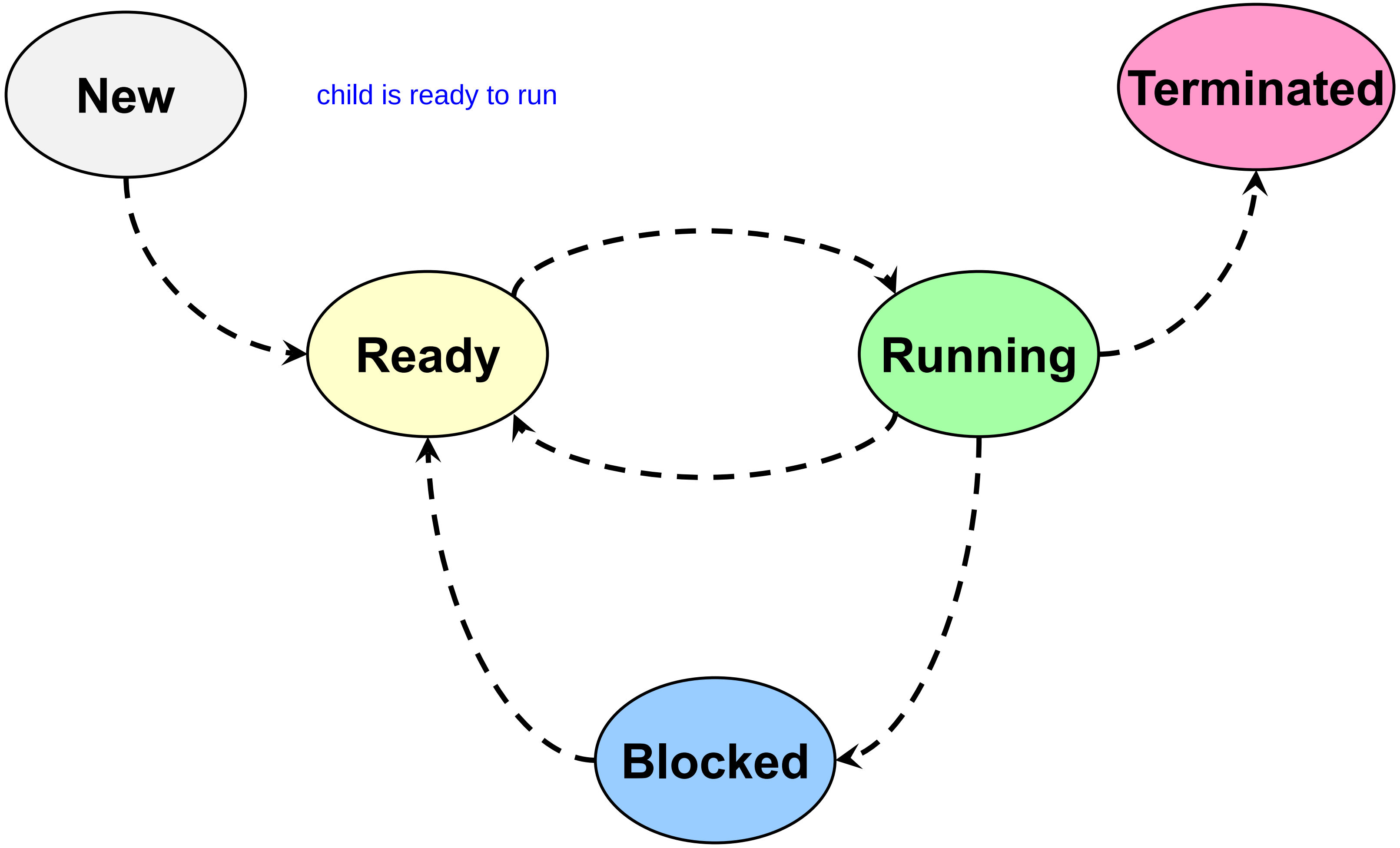


initially the child will only have read permissions - when an exception occurs this will mean that the child is going to write something and thus now there is a need to allocate new frame for this edited page

So, Effect of **fork**( )

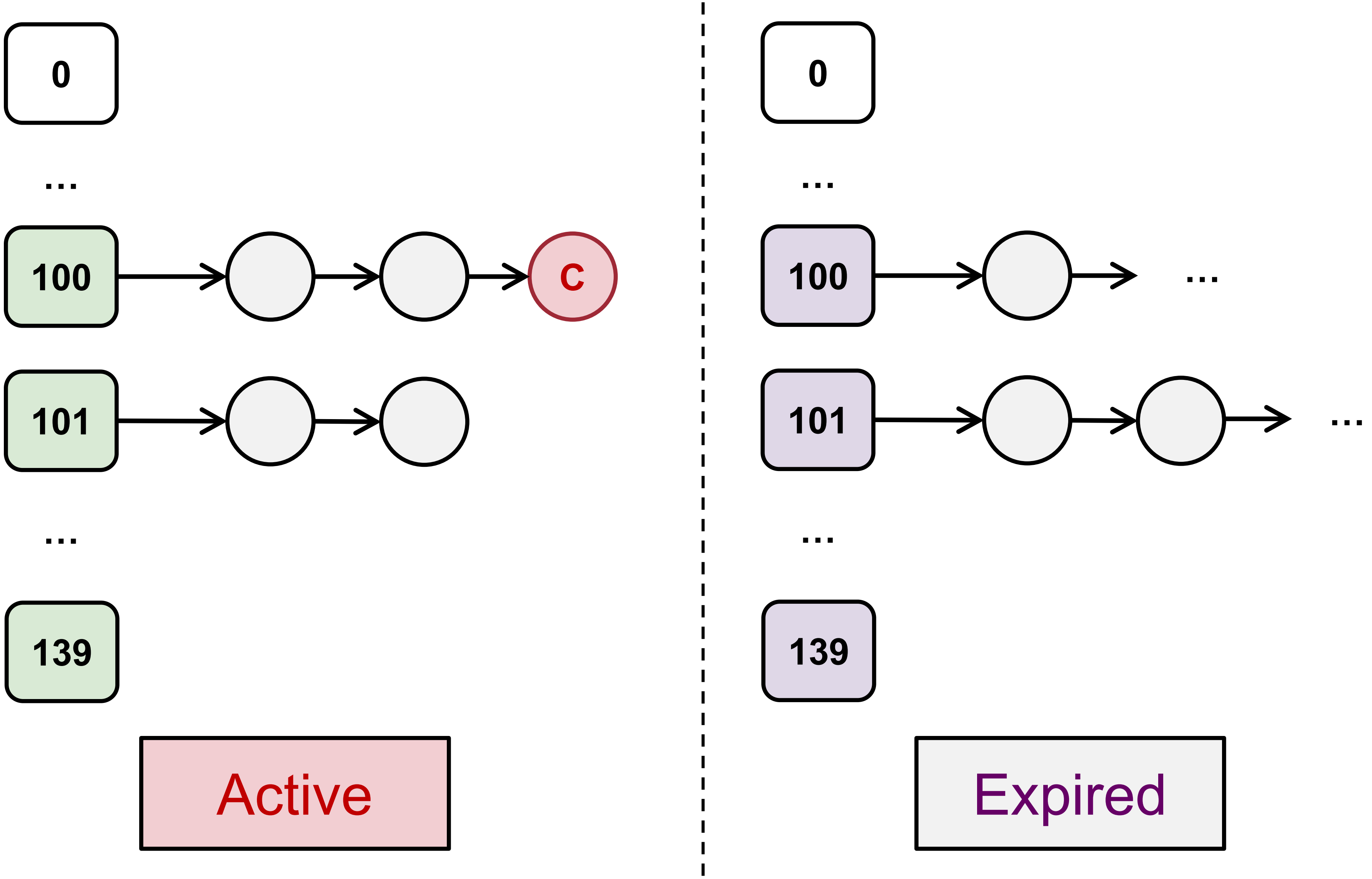


Child, welcome to the world!



Child, welcome to the Queue!

MLFQ means child might have the highest priority to run



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
}
```

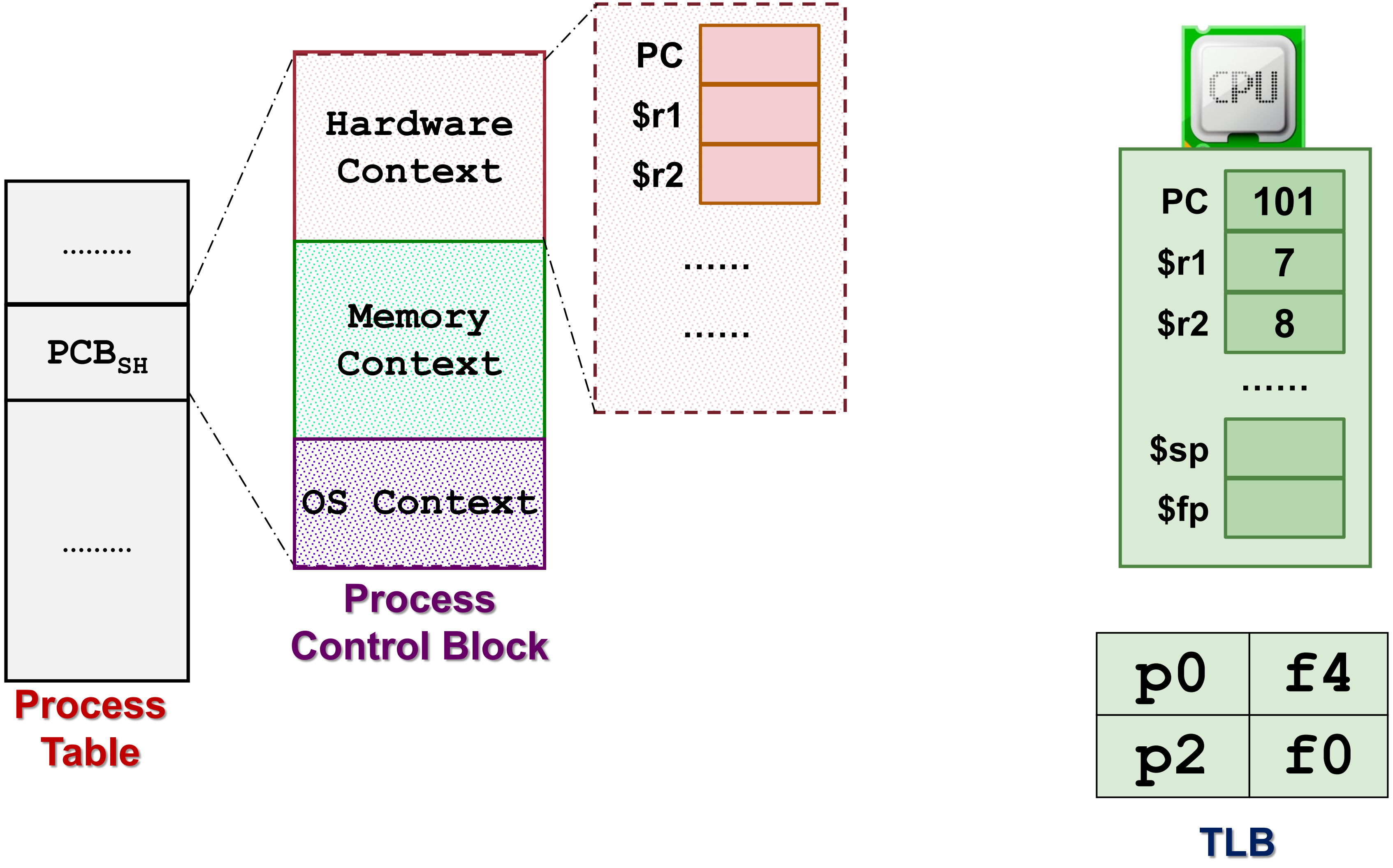
Kernel Mode

Typical steps for Shell Interpreter

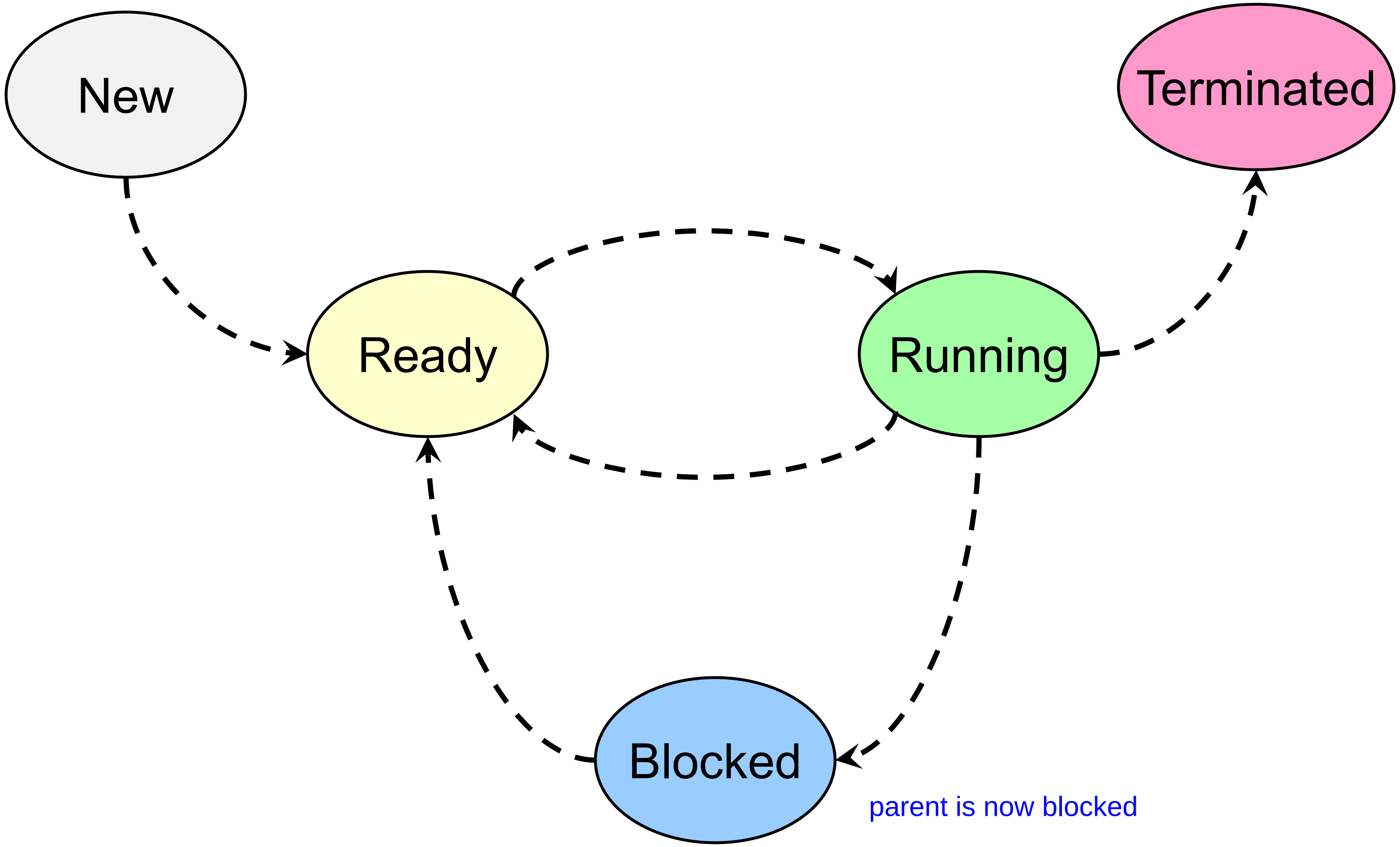
```
UserCmd  ← Read from keyboard
fork()
if I am the parent (i.e. the shell)
    wait ( child to finish )
else
    exec( UserCmd )
```

wait( ): The interpreter will….

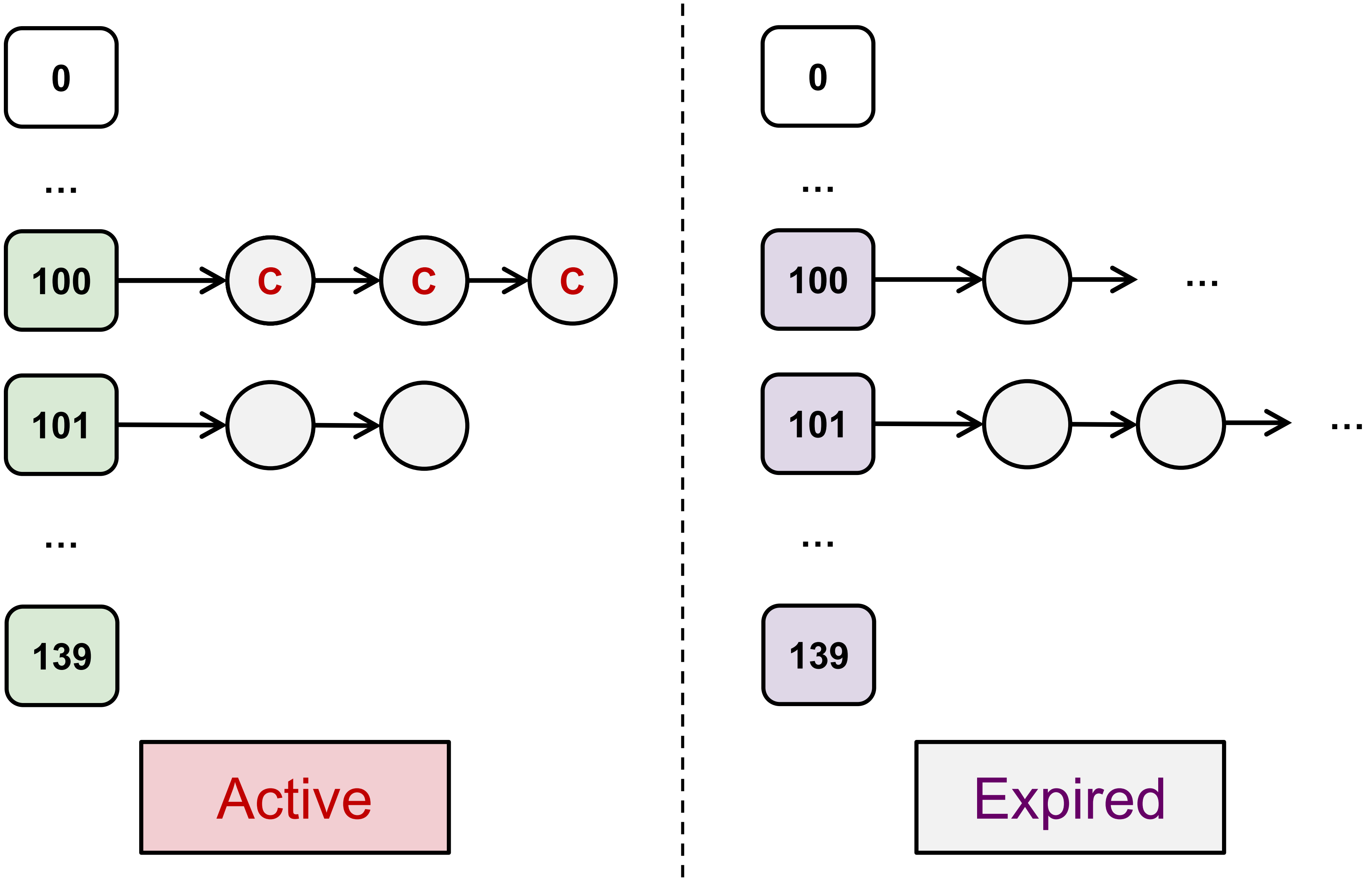
now the parent process will give up its processing time and save all its current information to the hardware context



The interpreter is now…

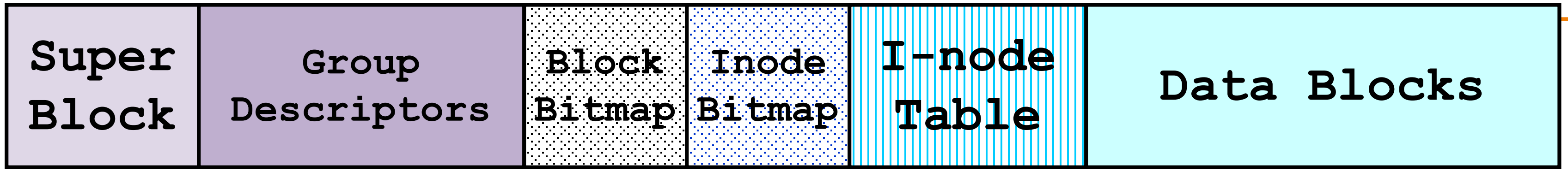


Hmm… who gets to run?

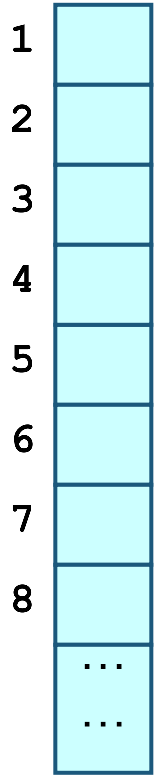


Typical steps for Shell Interpreter

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

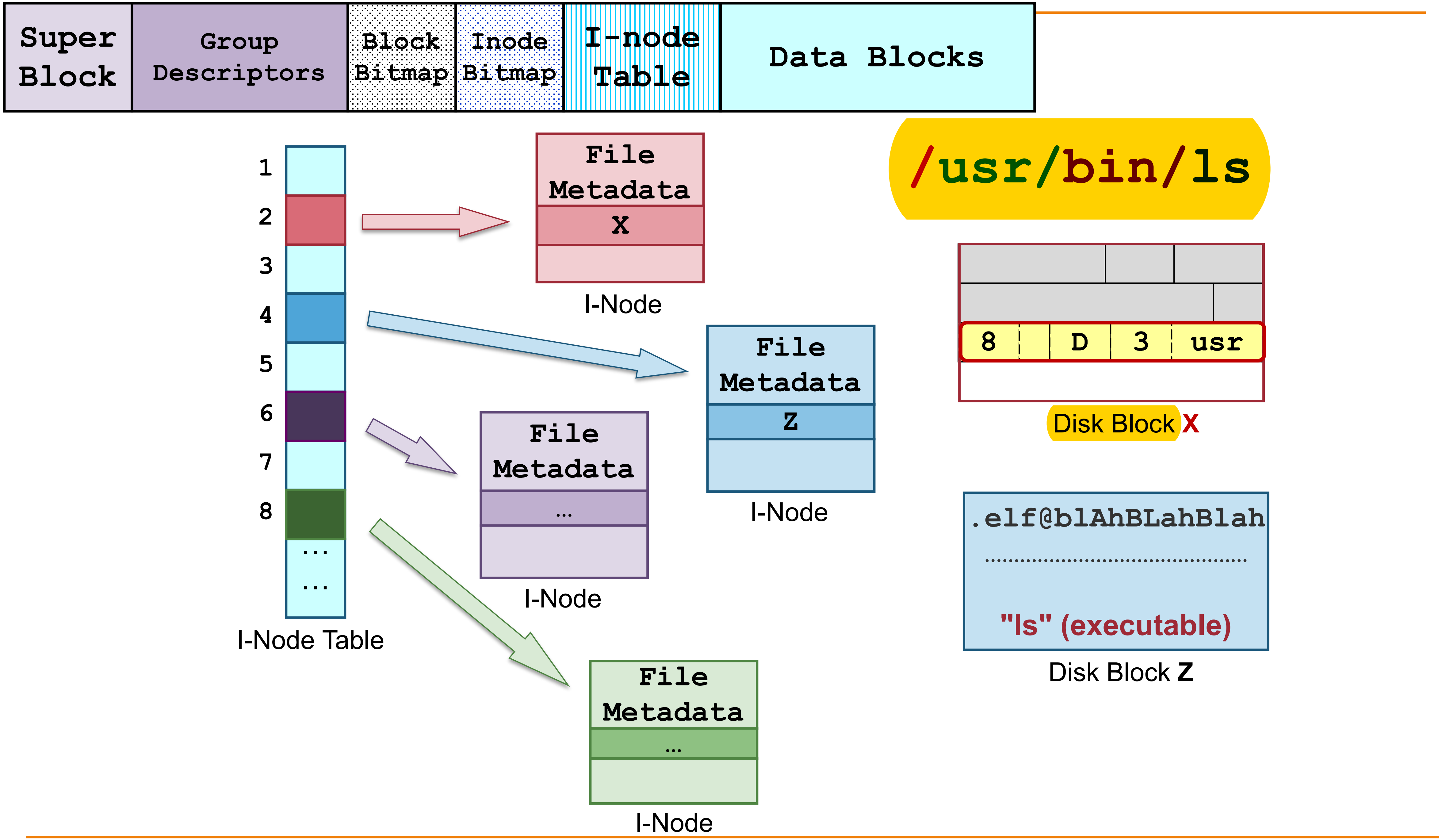


/usr/bin/ls



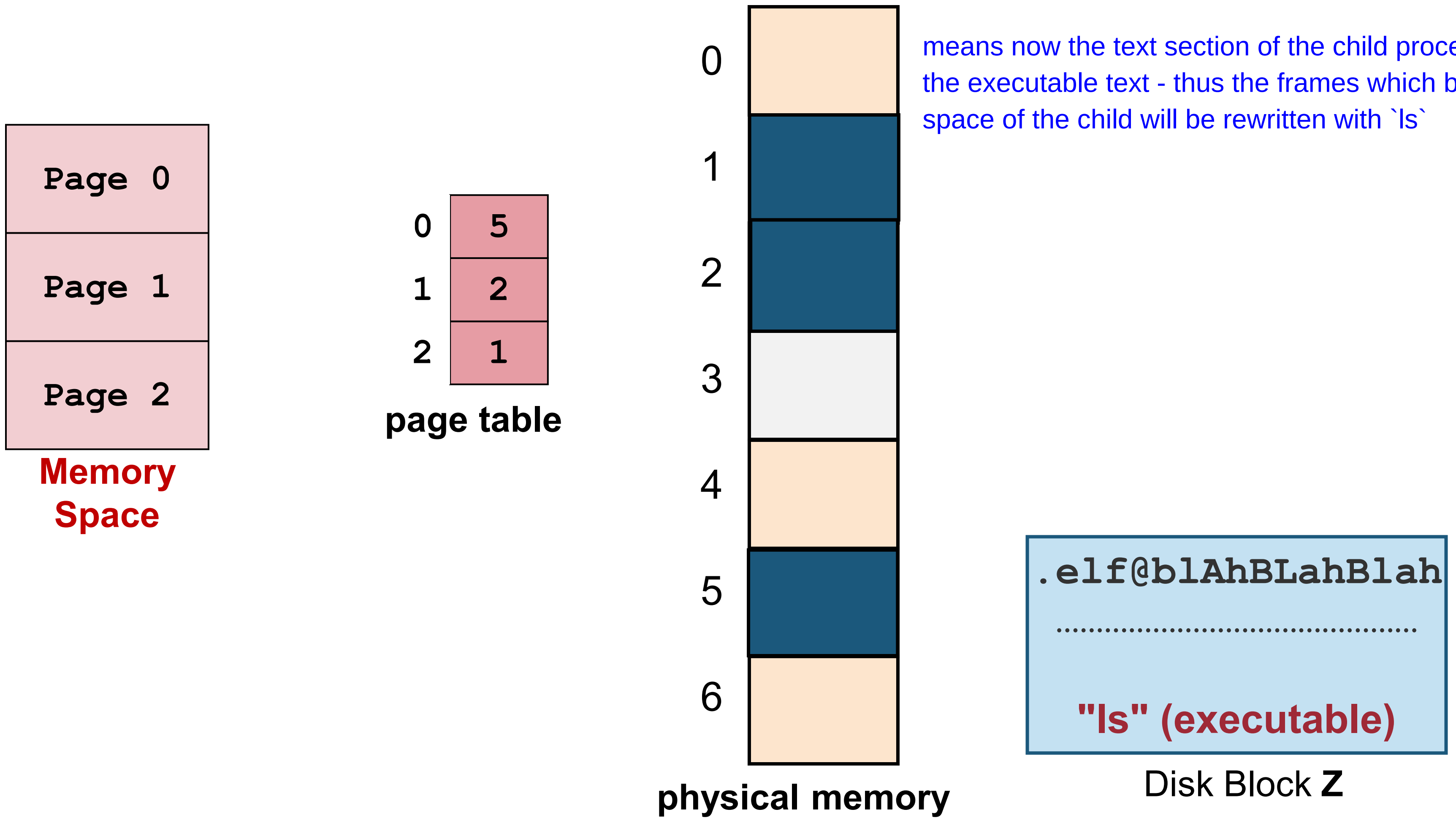
I-Node Table

FYI - inodes can be accessed even if its not from the same group - since each group will only have a fixed number of inodes



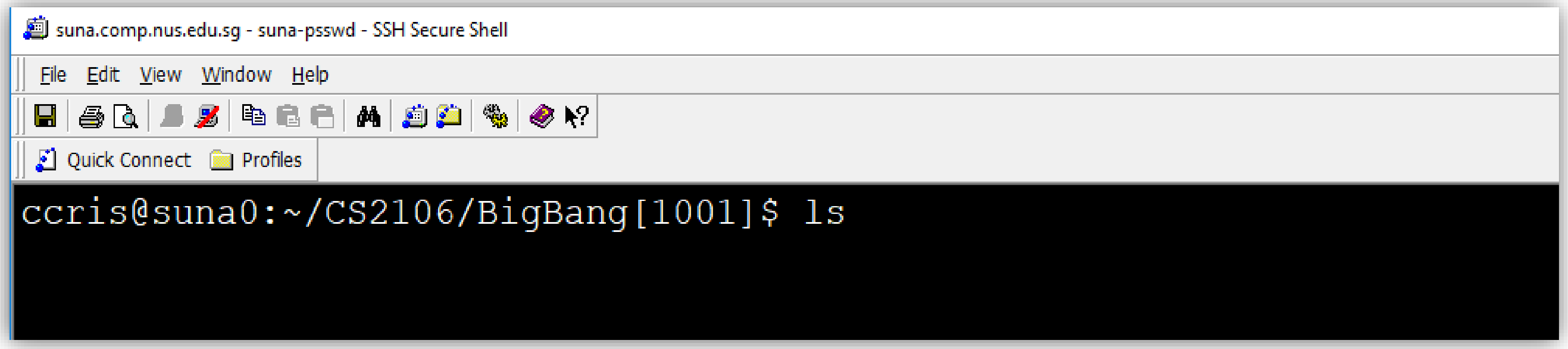
Super Block
Group Descriptors
Block Bitmap
Inode Bitmap
I-node Table
Data Blocks
/usr/bin/ls
1
2
3
4
5
6
7
8
...
...
I-Node Table
File Metadata
X
I-Node
File Metadata
Z
I-Node
File Metadata
...
I-Node
File Metadata
...
I-Node
8
D
3
usr
Disk Block X
.elf@blAhBLahBlah
"ls" (executable)
Disk Block Z

Memory Content Replaced

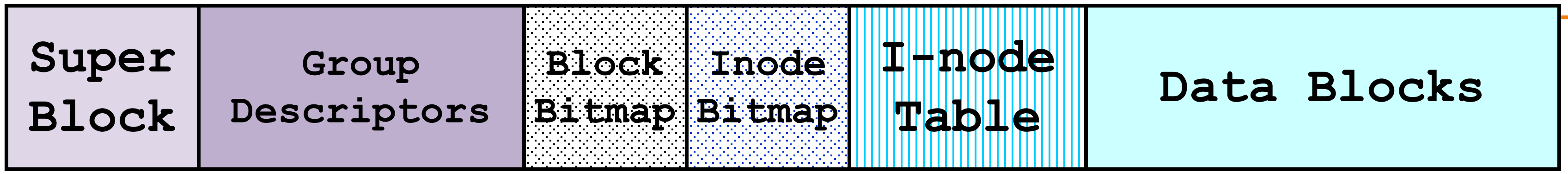


means now the text section of the child process will be replaced with the executable text - thus the frames which belong to the memory space of the child will be rewritten with `ls`

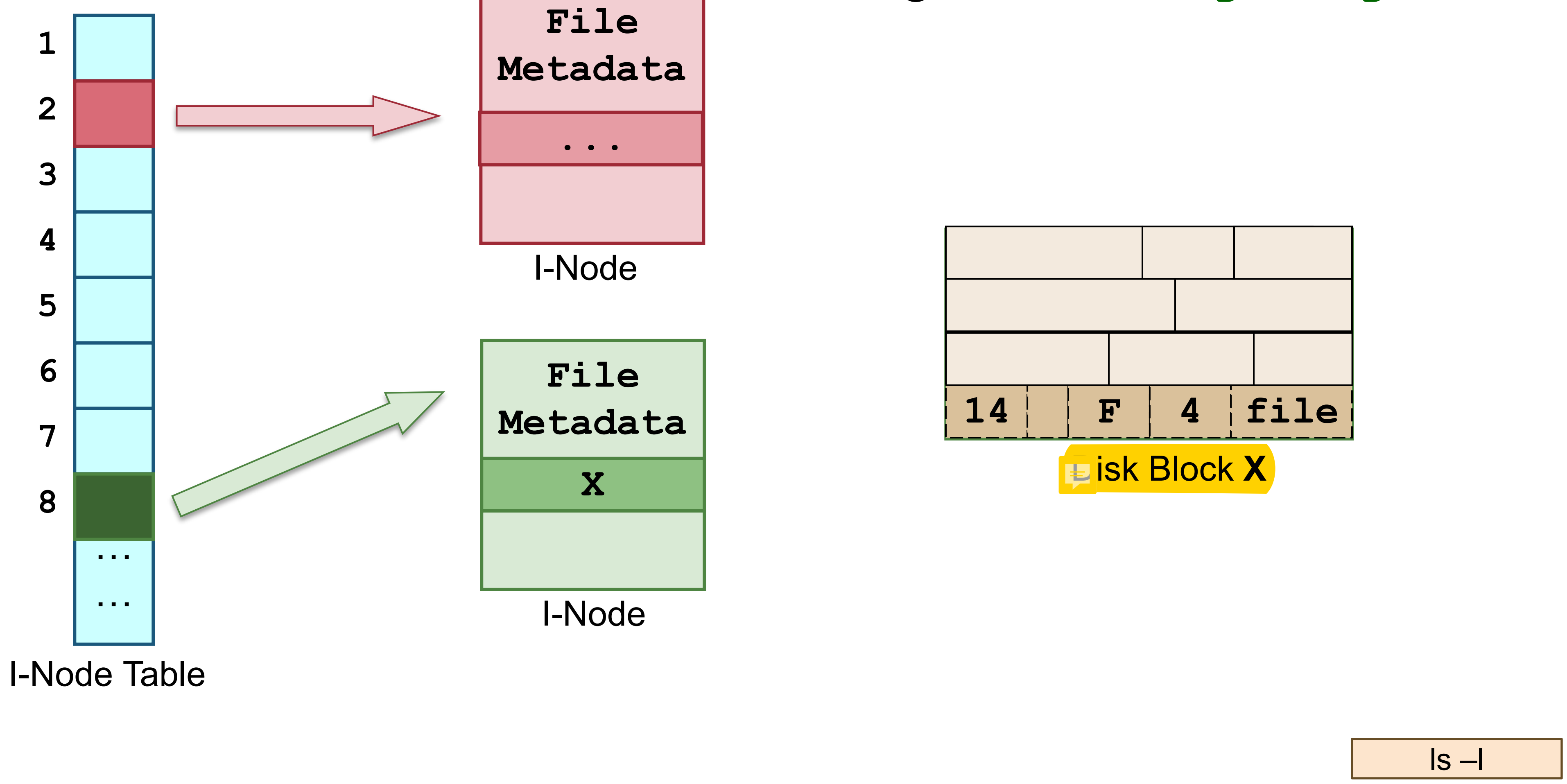
Child is now "ls", what next?



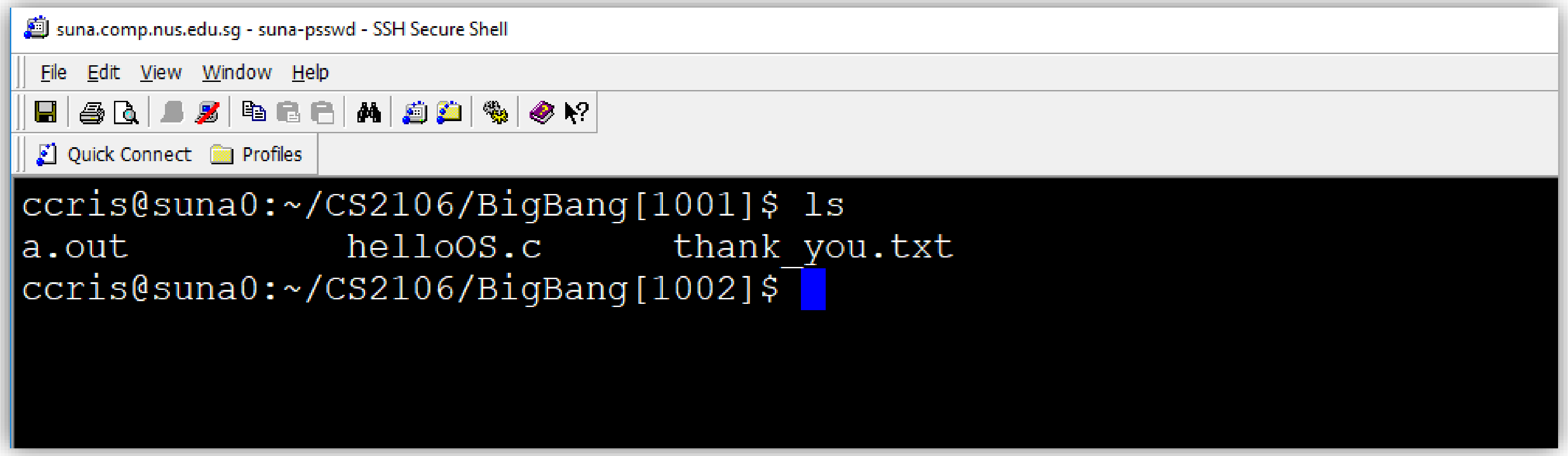
Listing **/…/…/BigBang**



Listing /…/…/BigBang

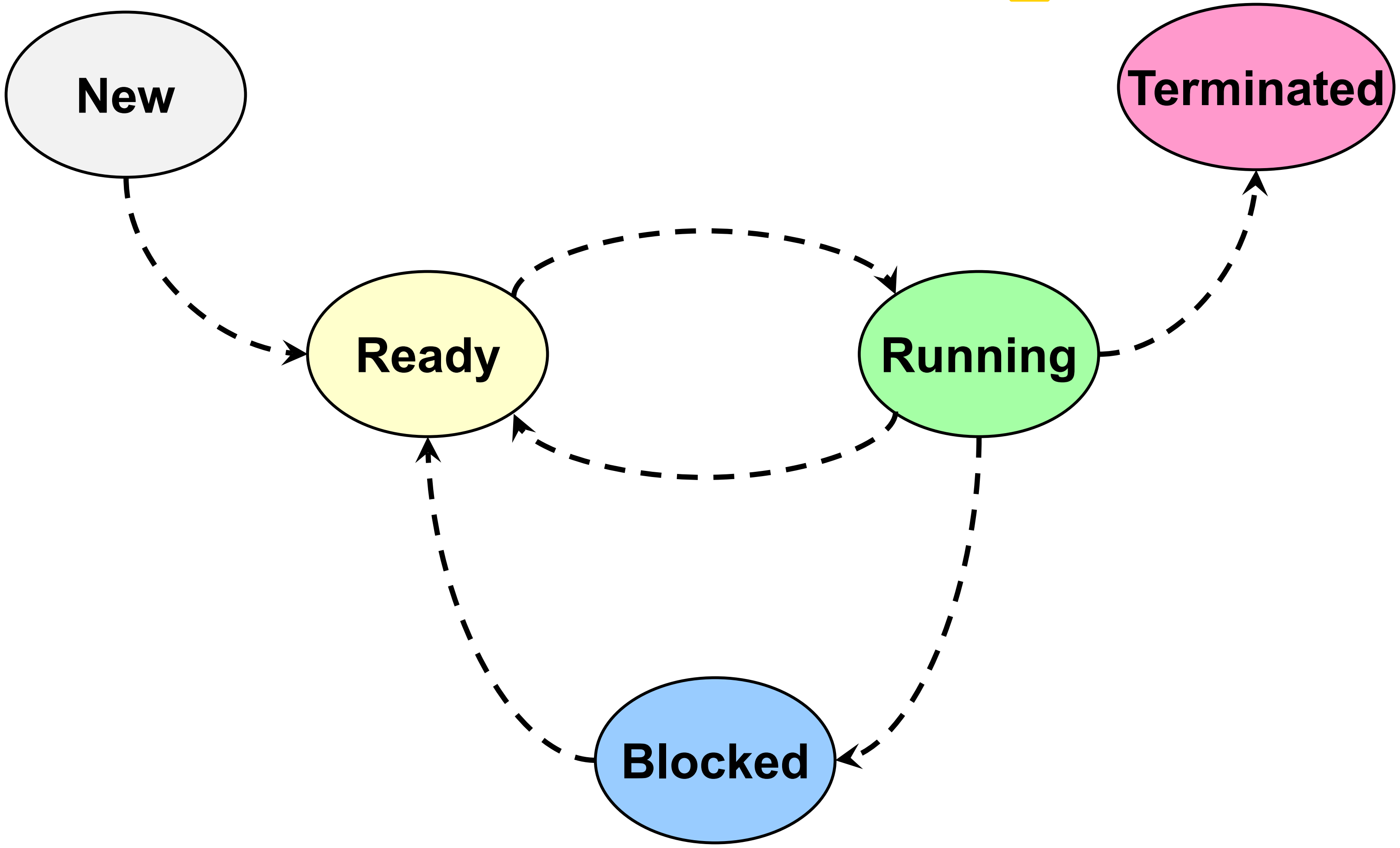


"ls" prints the directory content

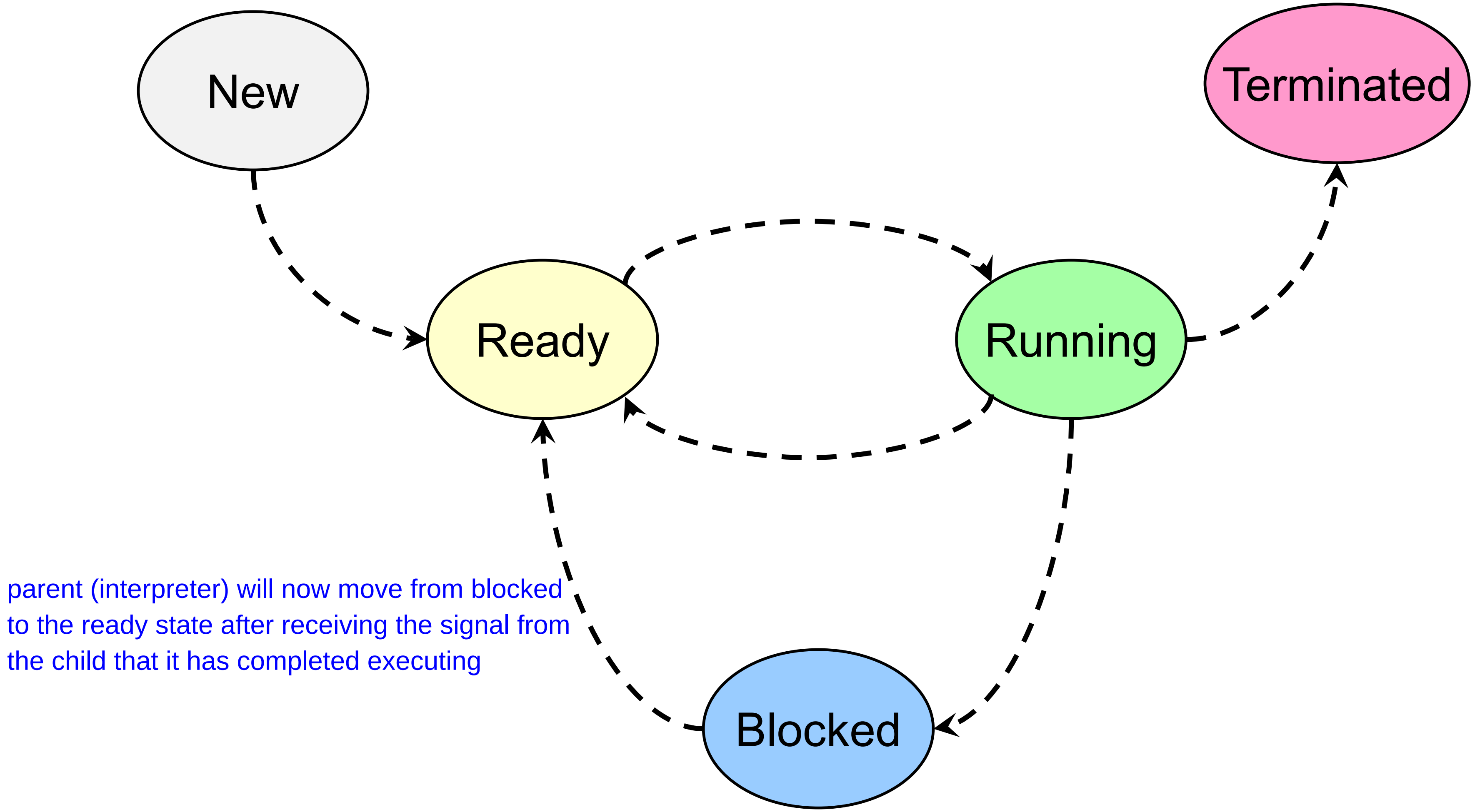


Child **exits**

child has finally terminated



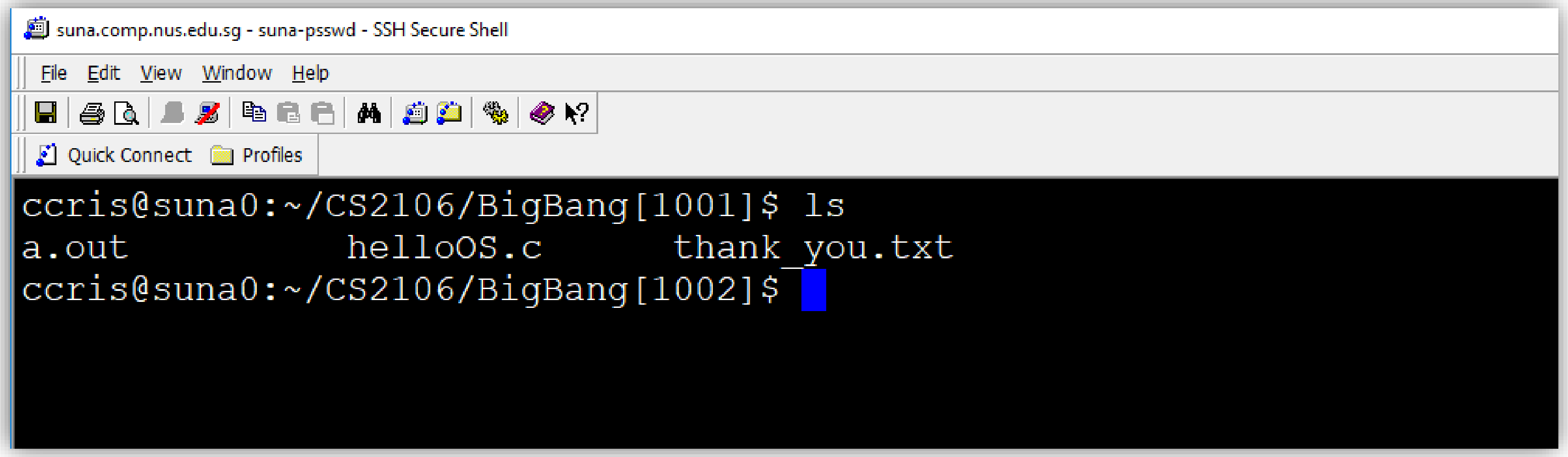
The interpreter **hears about it**….



Typical steps for Shell Interpreter

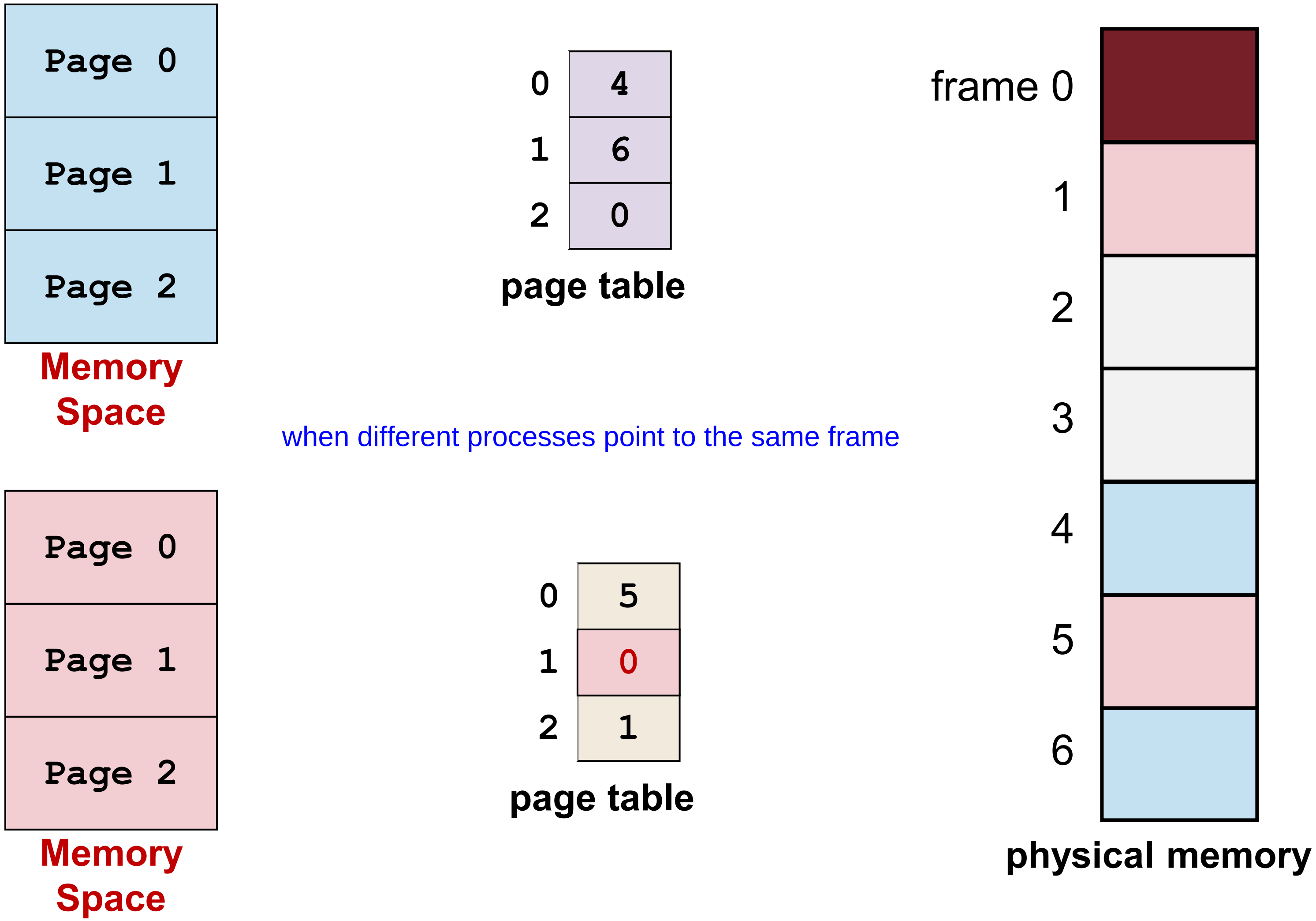
```
UserCmd  ← Read from keyboard
fork()
if I am the parent (i.e. the shell)
    wait ( child to finish )
else
    exec( UserCmd )
```

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]$

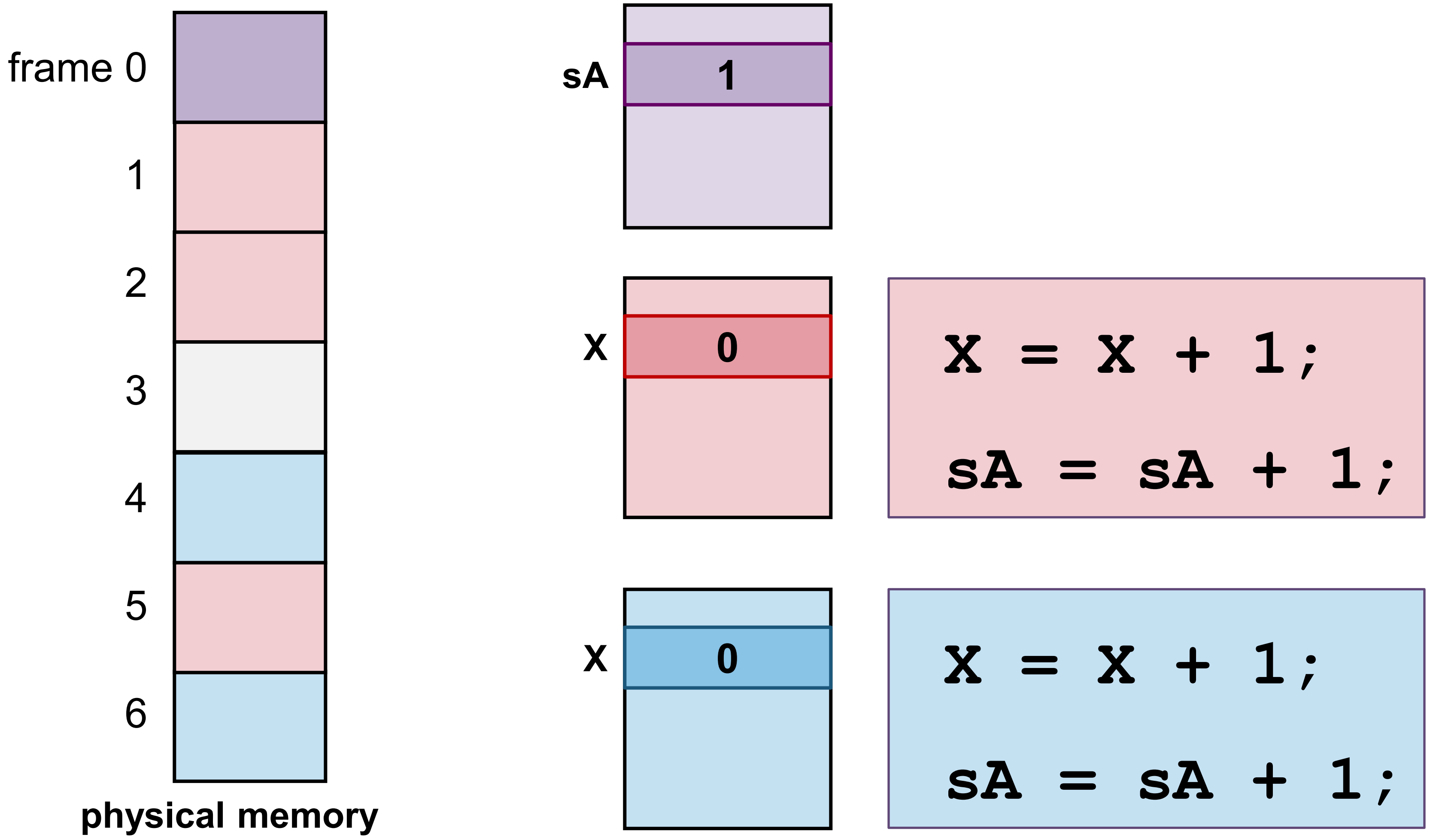
WE SHOULD SHARE!

How to **Share Memory**?



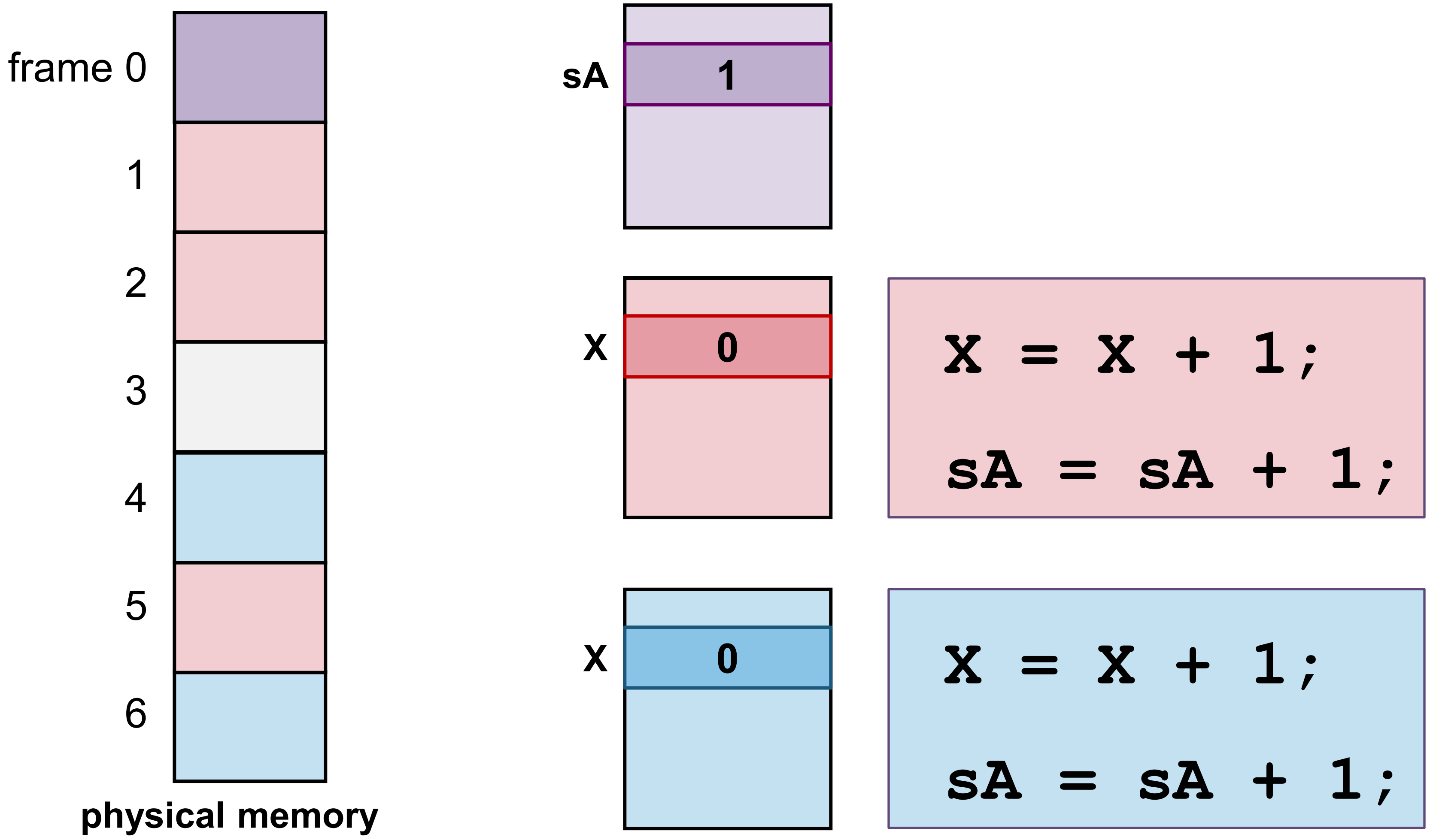
when different processes point to the same frame

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



Semaphore to the rescue!

semaphores create a critical section to ensure that access to a shared memory is done in a sequence



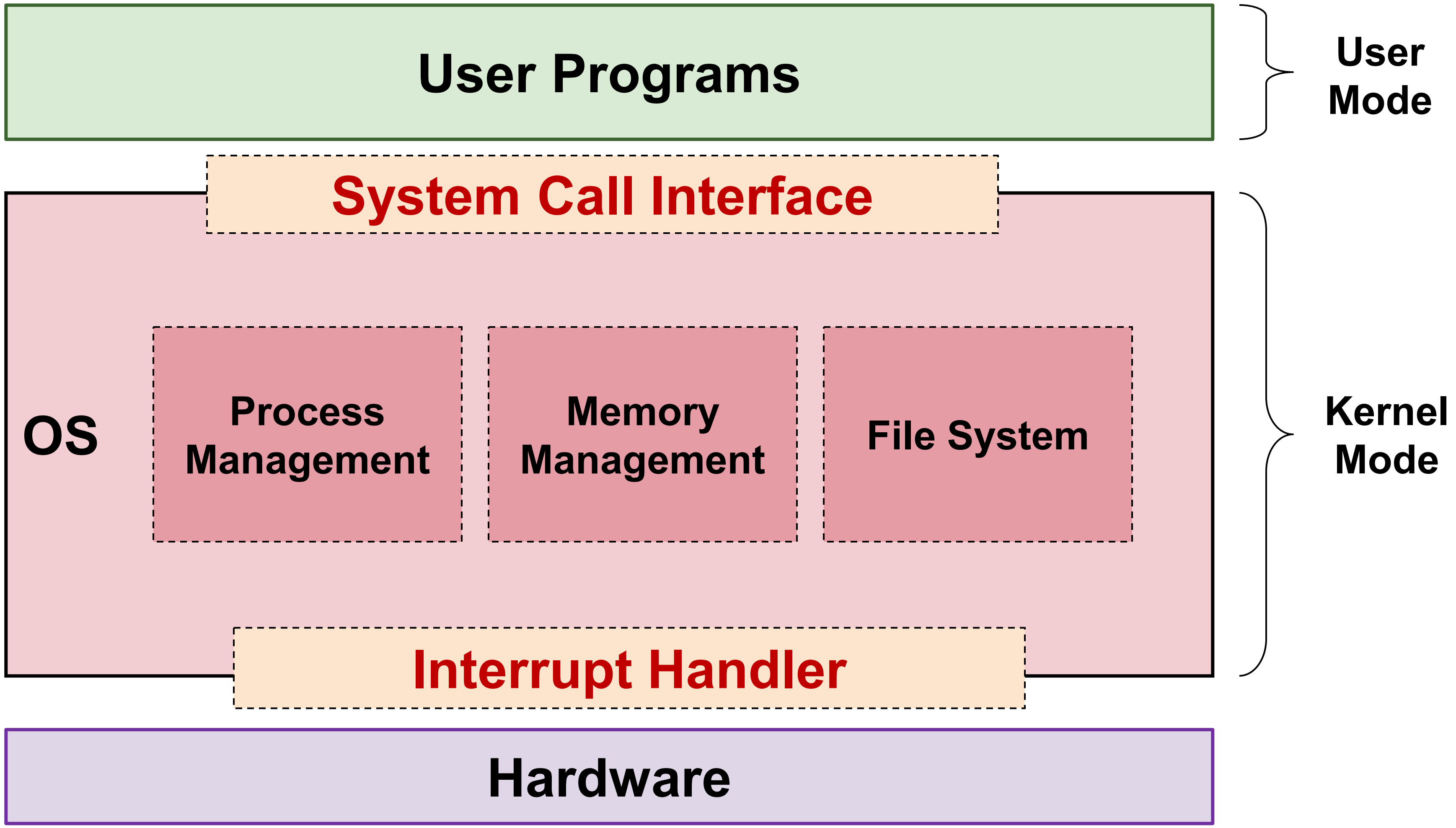
Concurrency

- 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!