Tutorial 2

ECSE 420 – Tutorial 2

Dimitrios Stamoulis

TR 4110
September 29, 2014

ECSE 420 – Tutorials (1/2)

- Hours and Location
  - Group 1 : Mondays,   04:00 PM - 5:30 PM (TR4110)
  - Group 2 : Wednesdays, 04:00 PM - 5:30 PM (TR4110)

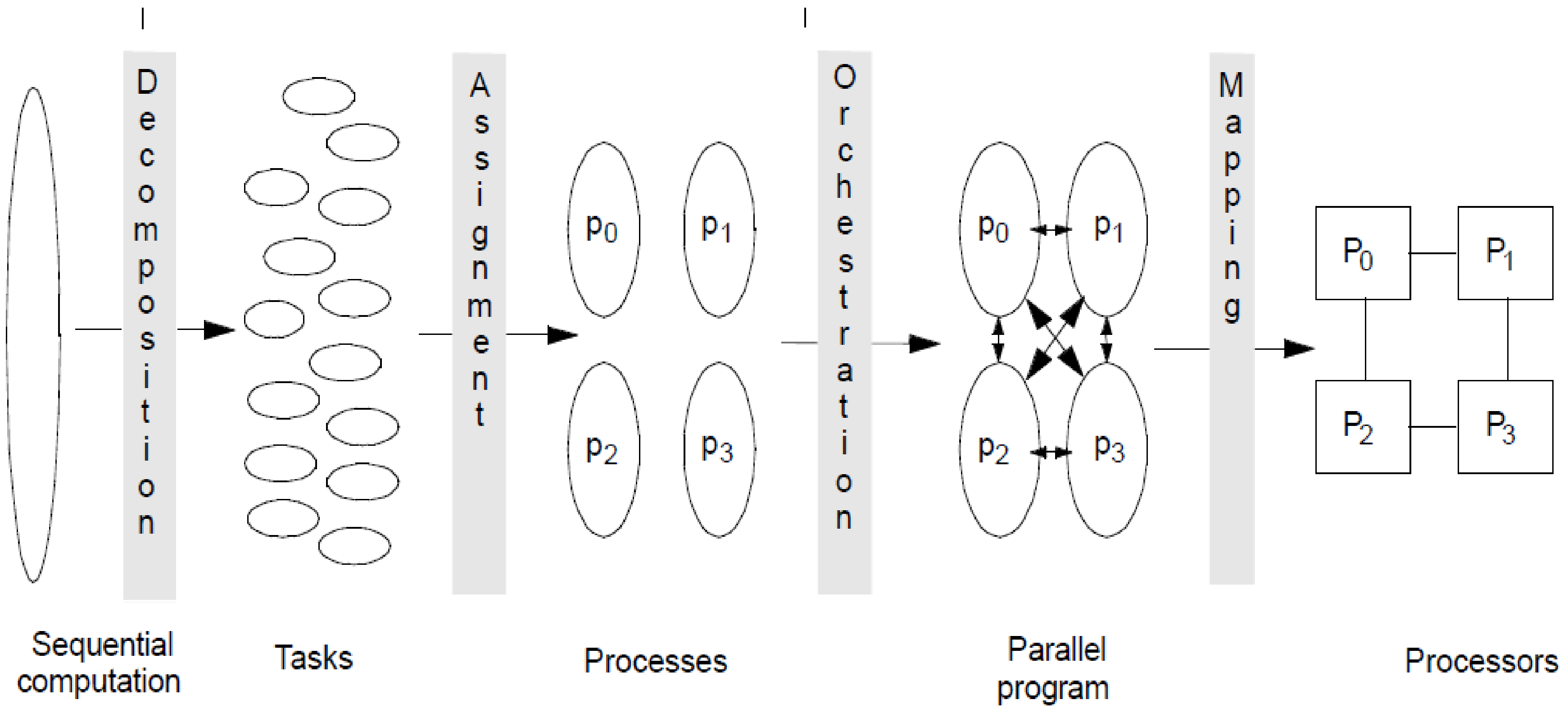
- Our goal to have a foretaste of:
  - ➢ Useful parallel programming tools (e.g OpenMP, MPI)
  - ➢ Midterm exercises, assignments etc

- So, please ask questions !!
  - ➢ Office Hours :
    Wednesdays, 10.00-11.30am, McConnell 544
  - ➢ Mail them @ :
    dimitrios.stamoulis@mail.mcgill.ca

ECSE 420 – Tutorials (2/2)

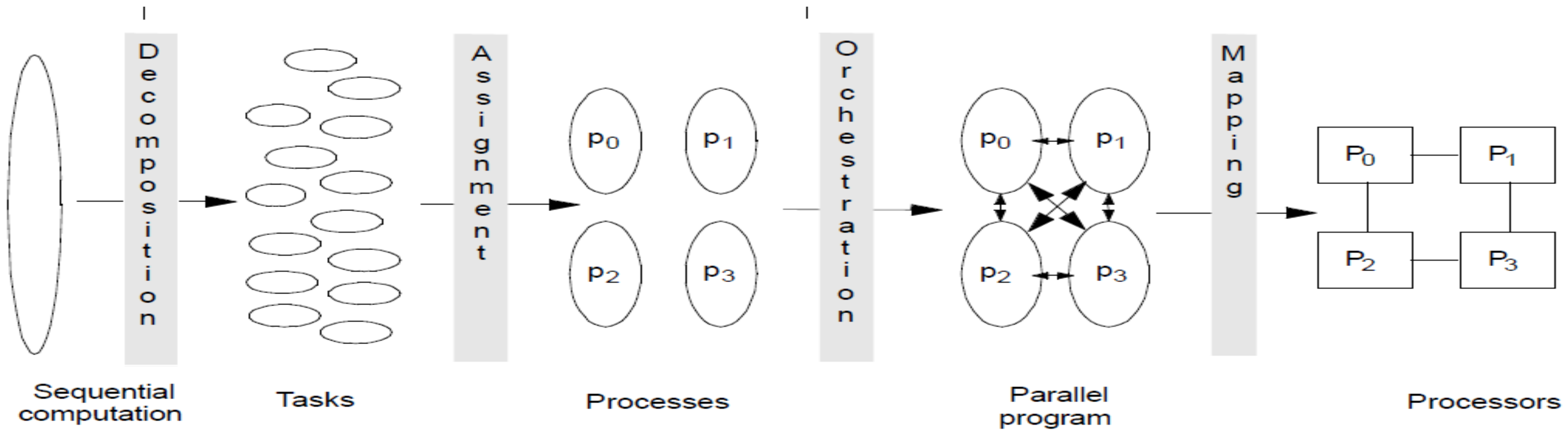
- Labs Schedule (tentative)
  - Tutorial 1 - 09/22 :
    Processes & Threads (Introduction)
  - **Tutorial 2 - 09/29 :**
    Processes & Threads (Assignment 1 & Lab 1)
  - Tutorial 3 - 10/06 :
    Shared Memory/ Msg Passing (Assignment 2)
  - Tutorial 4 - 10/13 :
    Shared Memory/ Msg Passing (Lab 2)
  - Tutorial 5 - 10/20 :
    Parallel/ Distributed Program. (Assignment 3)
  - Tutorial 6 – 10/27:
    Parallel/ Distributed Program. (Lab 3)

Tutorial 2 – Main Programming scheme



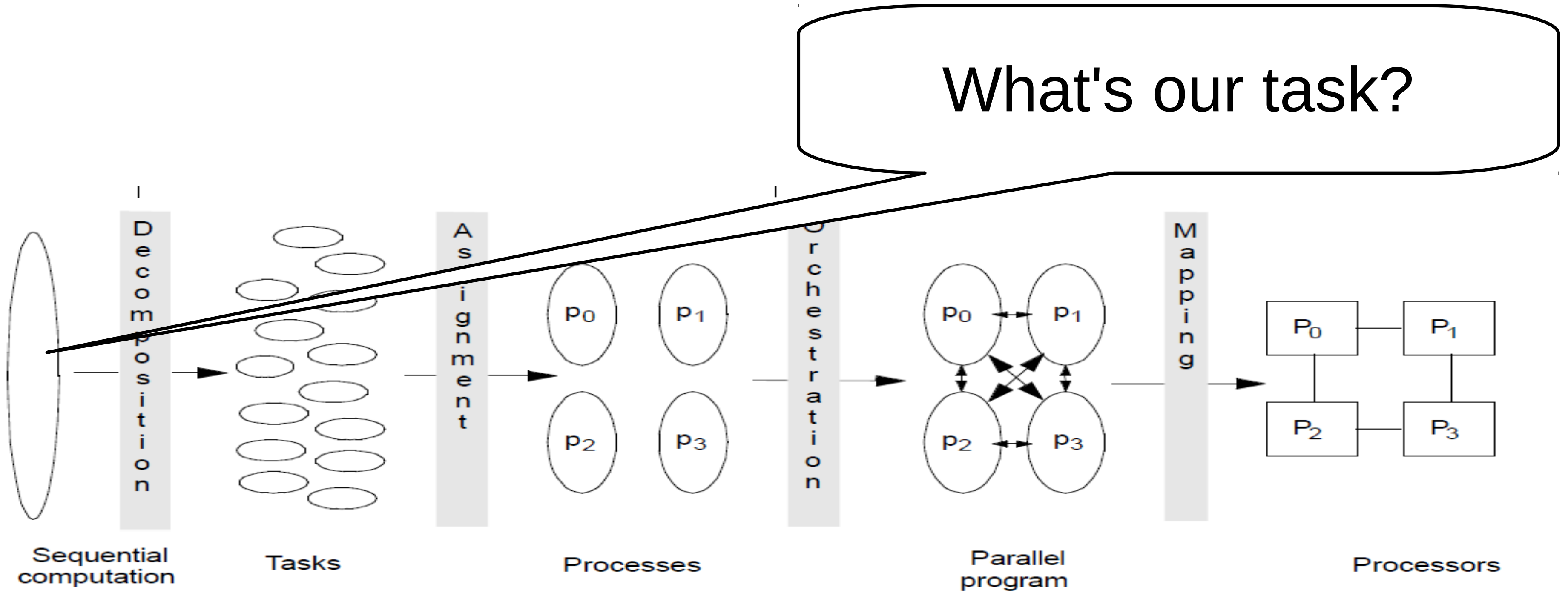
That is what we will follow for Lab 1..!!

Introduction to Lab 1



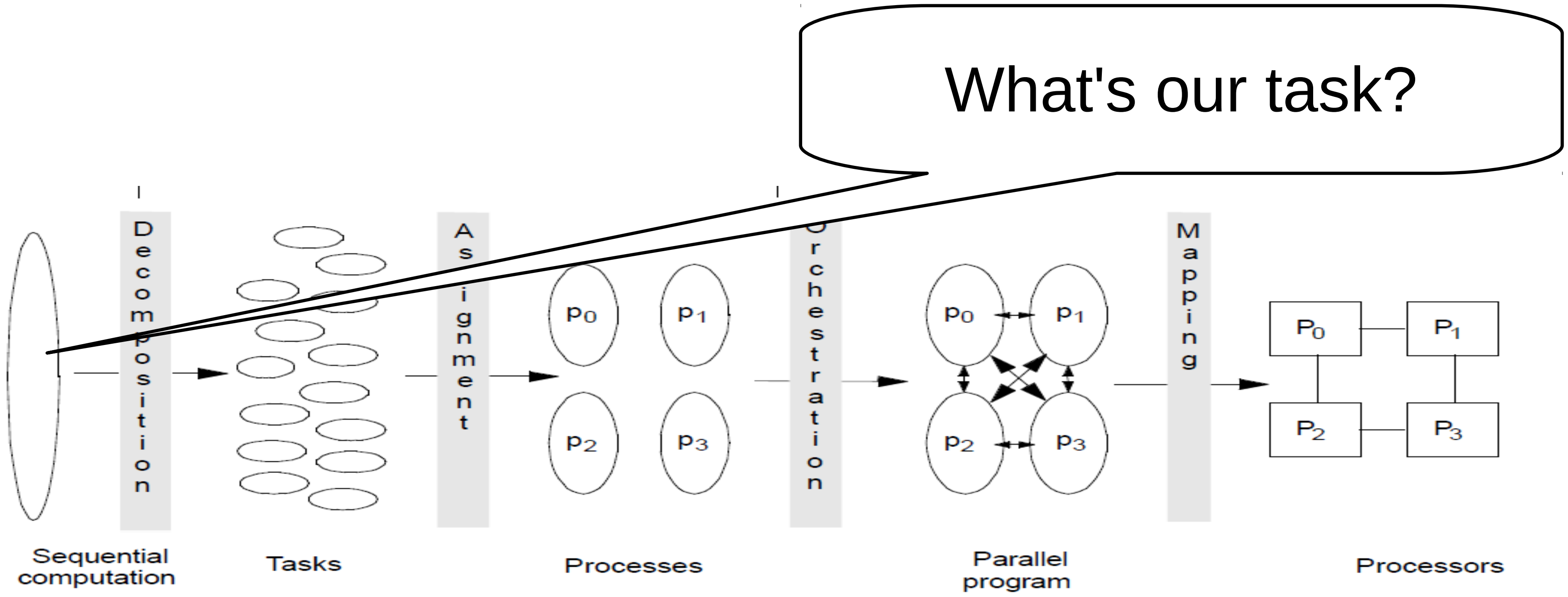
1) …
2) …
3) …
4) …

Introduction to Lab 1



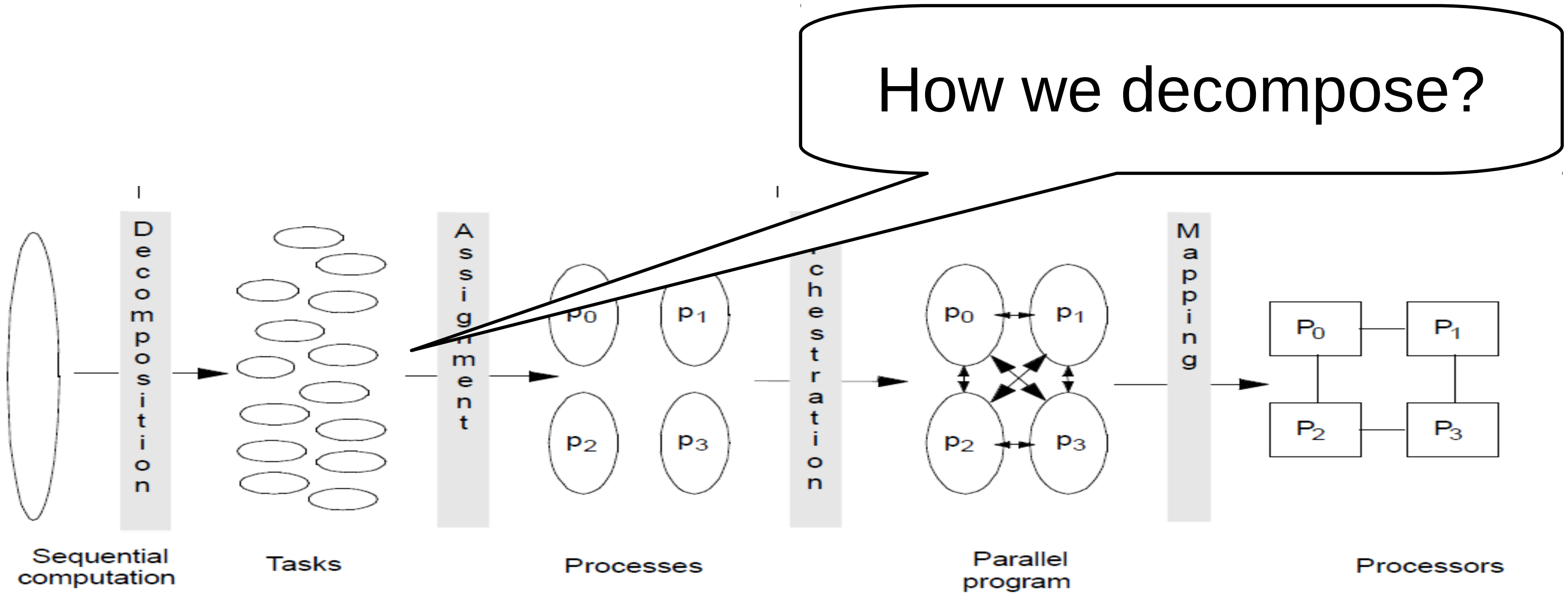
1) ...
2) ...
3) ...
4) ...

Introduction to Lab 1



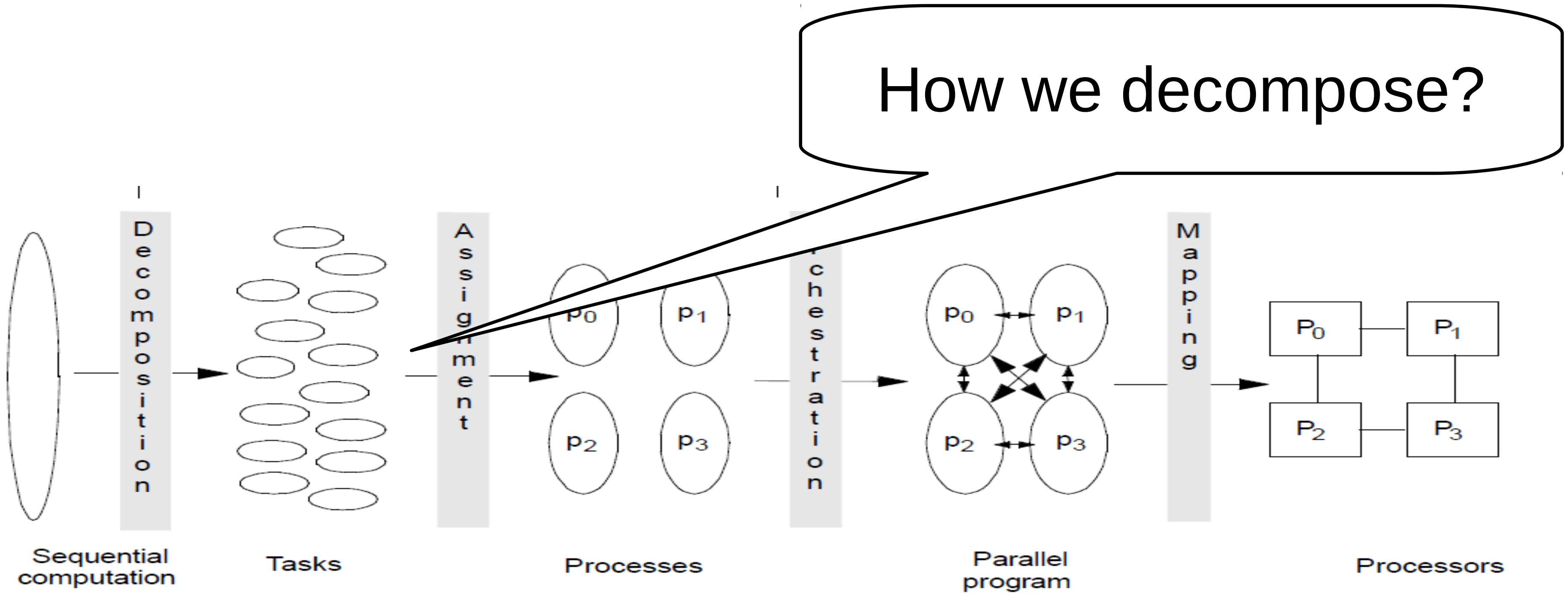
1) Task to be performed – Image processing
2) ...
3) ...
4) ...

Introduction to Lab 1



1) Task to be performed – Image processing
2) ...
3) ...
4) ...

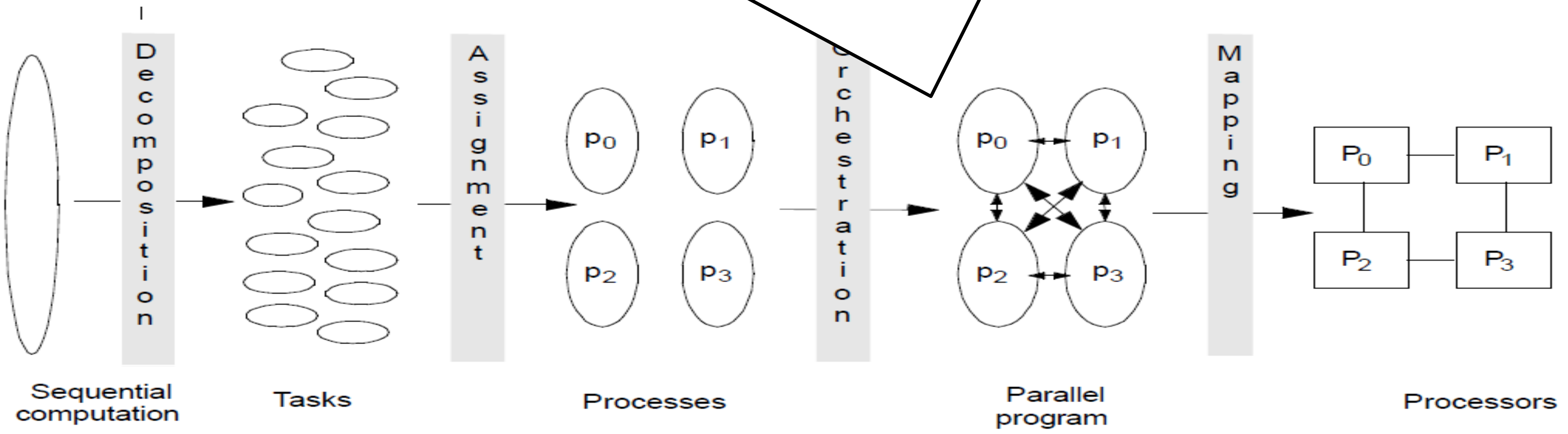
Introduction to Lab 1



1) Task to be performed – Image processing
2) Parallelization scheme – Processes
3) ...
4) ...

Introduction to Lab 1

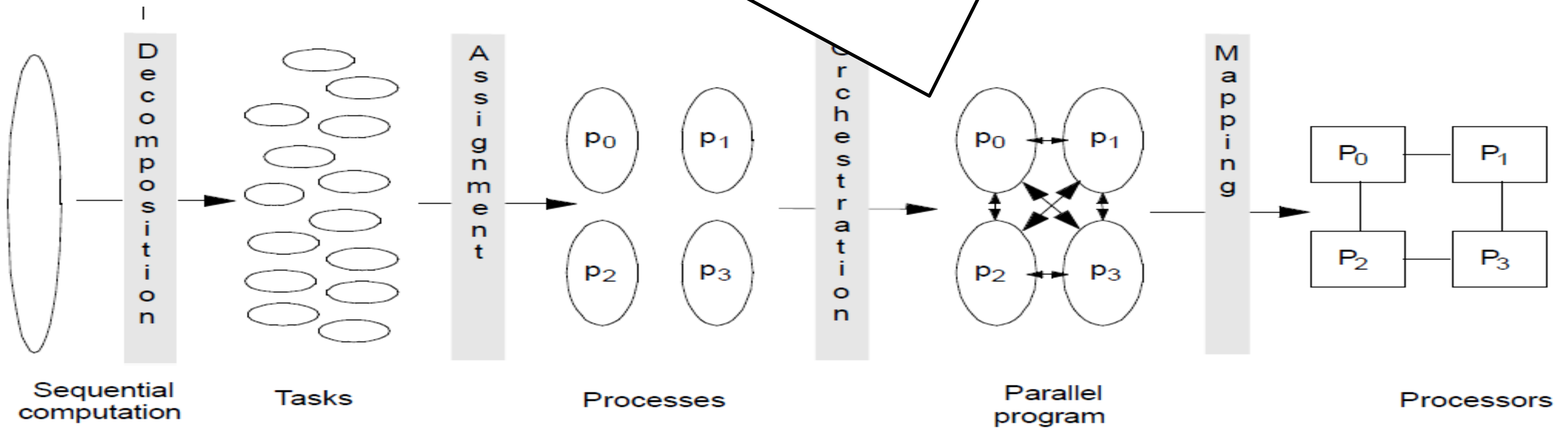
What should be synchronized?



1) Task to be performed – Image processing
2) Parallelization scheme – Processes
3) ...
4) ...

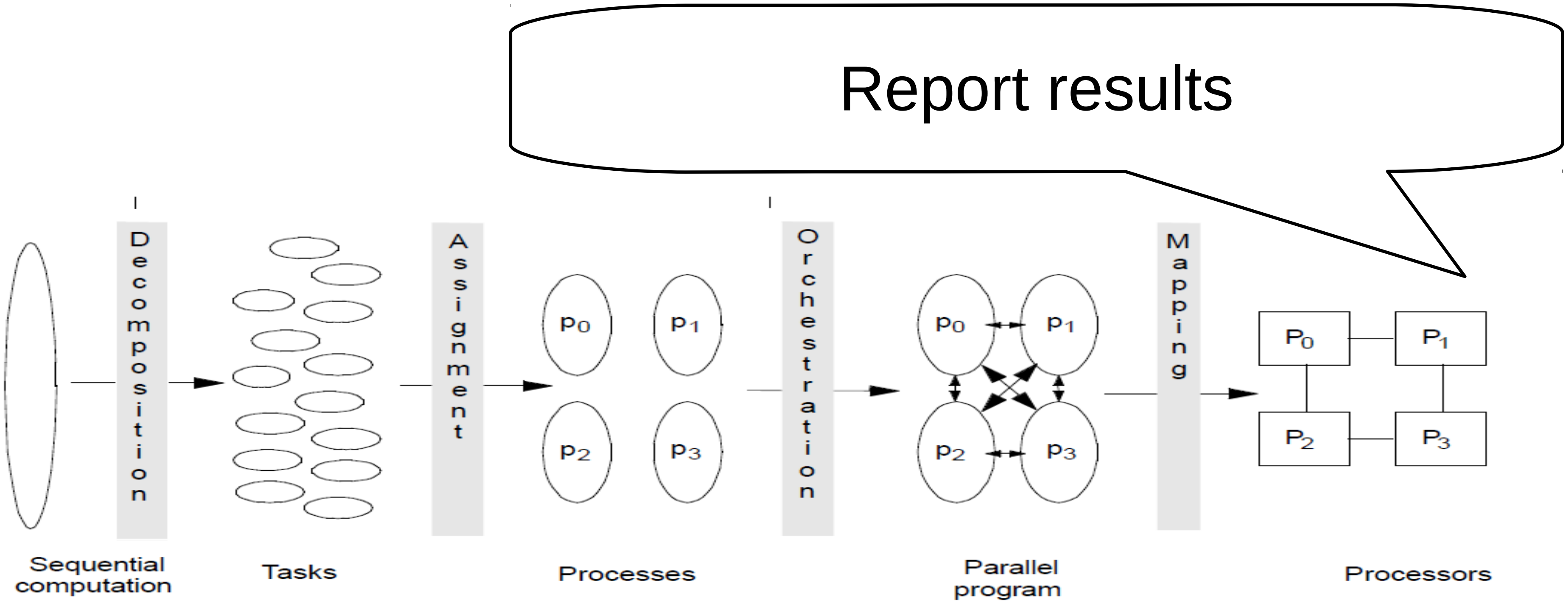
Introduction to Lab 1

What should be synchronized?



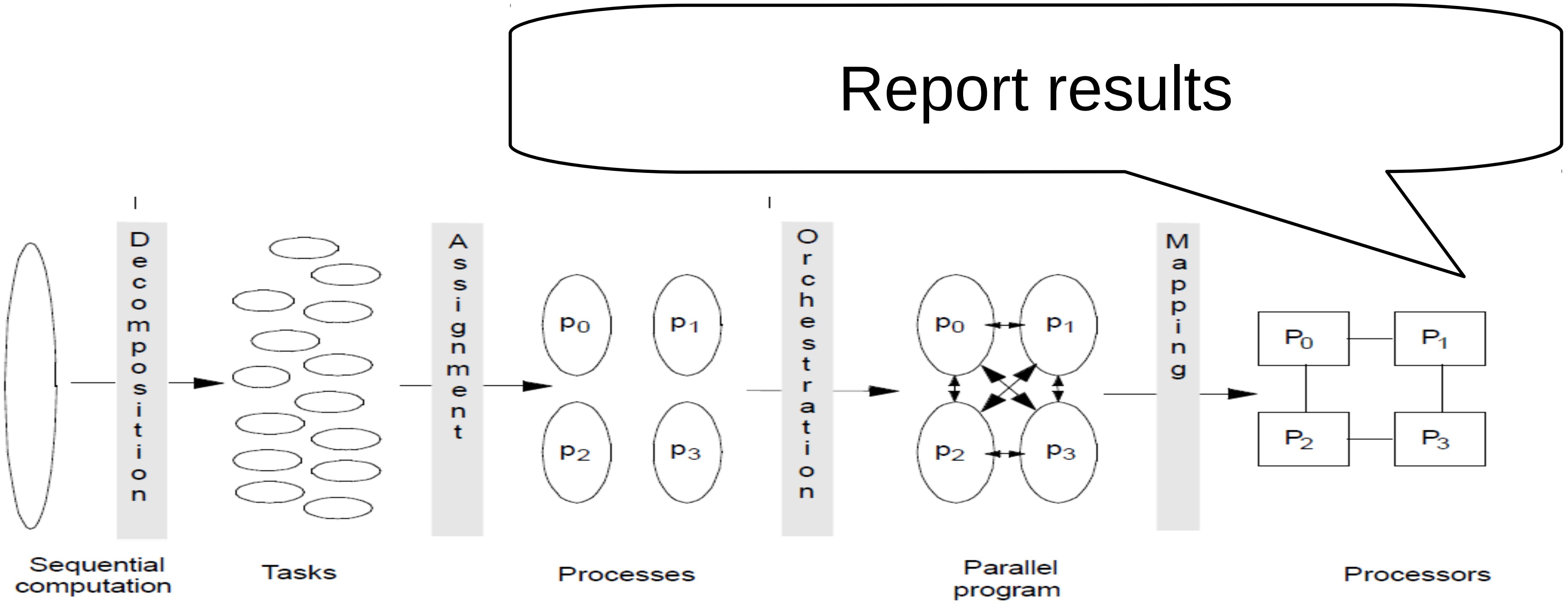
1) Task to be performed – Image processing
2) Parallelization scheme – Processes
3) Synchronization scheme – Pipes
4) ...

Introduction to Lab 1



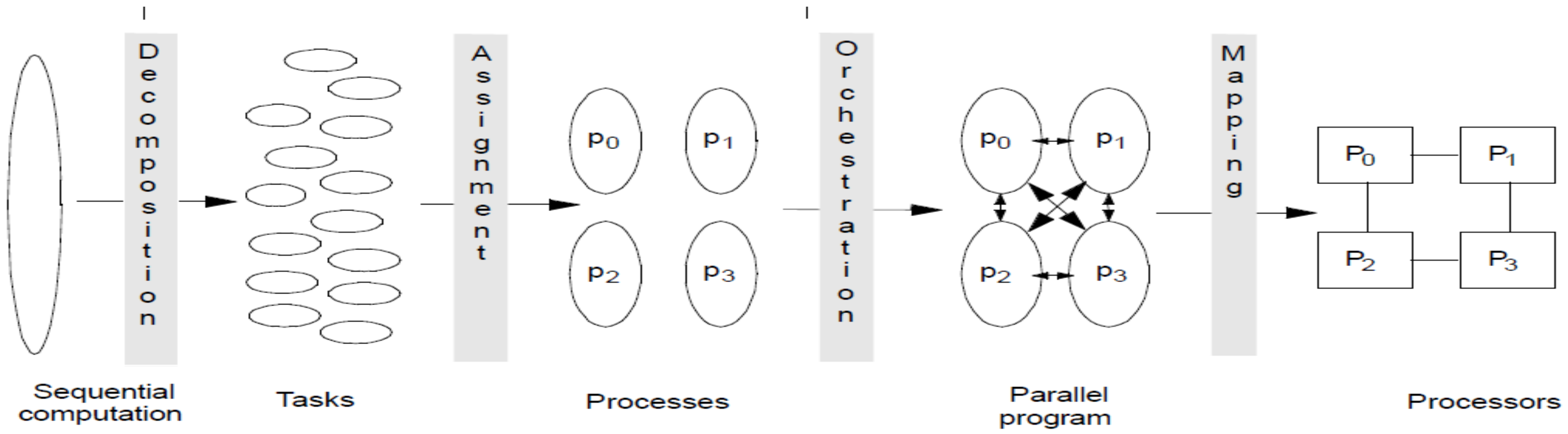
1) Task to be performed – Image processing
2) Parallelization scheme – Processes
3) Synchronization scheme – Pipes
4) ...

Introduction to Lab 1



1) Task to be performed – Image processing
2) Parallelization scheme – Processes
3) Synchronization scheme – Pipes
4) Report results

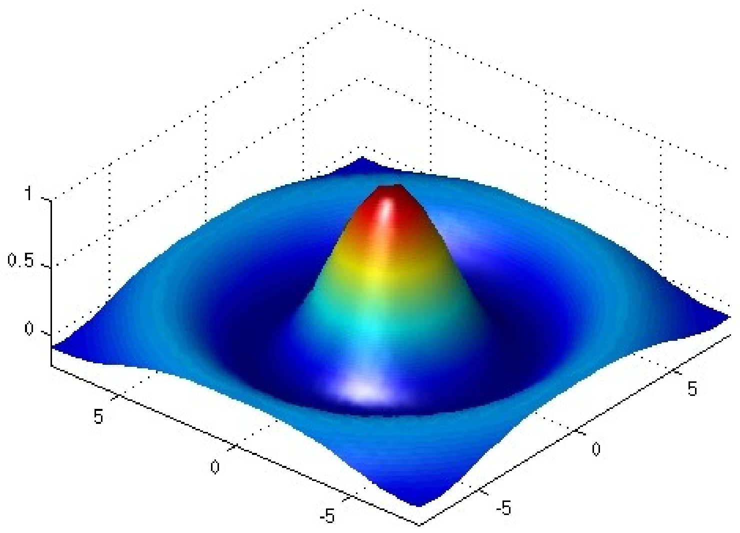
Introduction to Lab 1

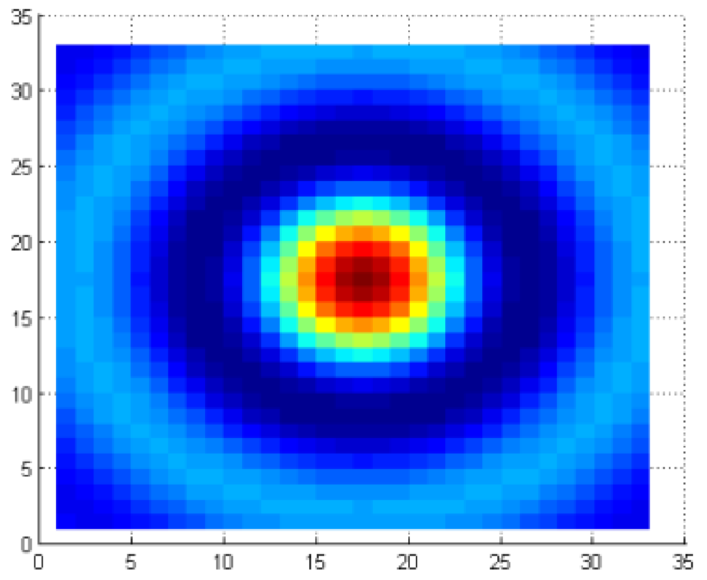


1) **Task to be performed – Image processing**
2) Parallelization scheme – Processes
3) Synchronization scheme – Pipes
4) Report results

Part 1 – Image processing

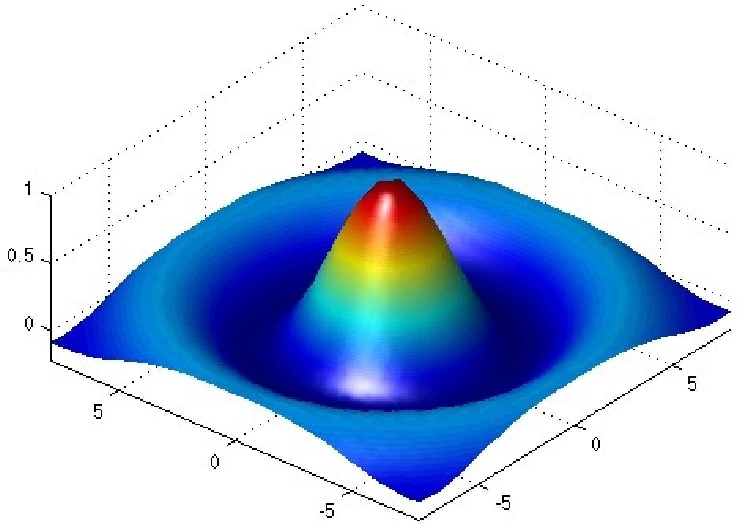
- Task to be performed = Image processing operator

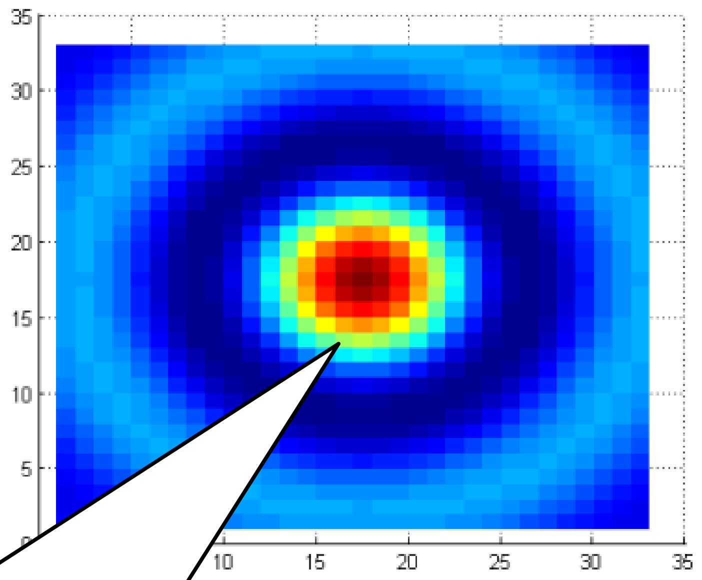




Part 1 – Image processing

- Task to be performed = Image processing operator

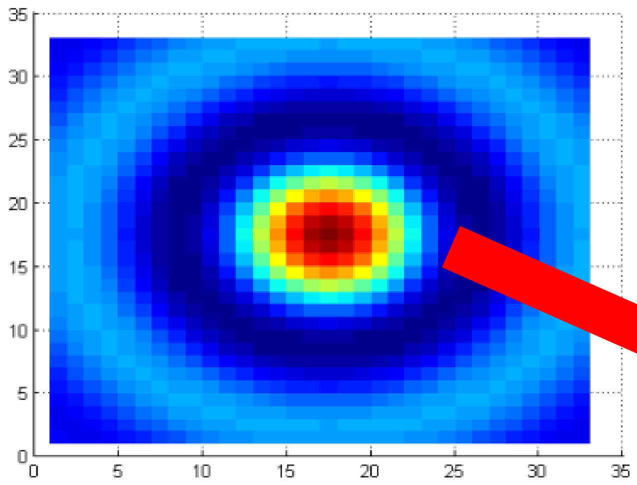


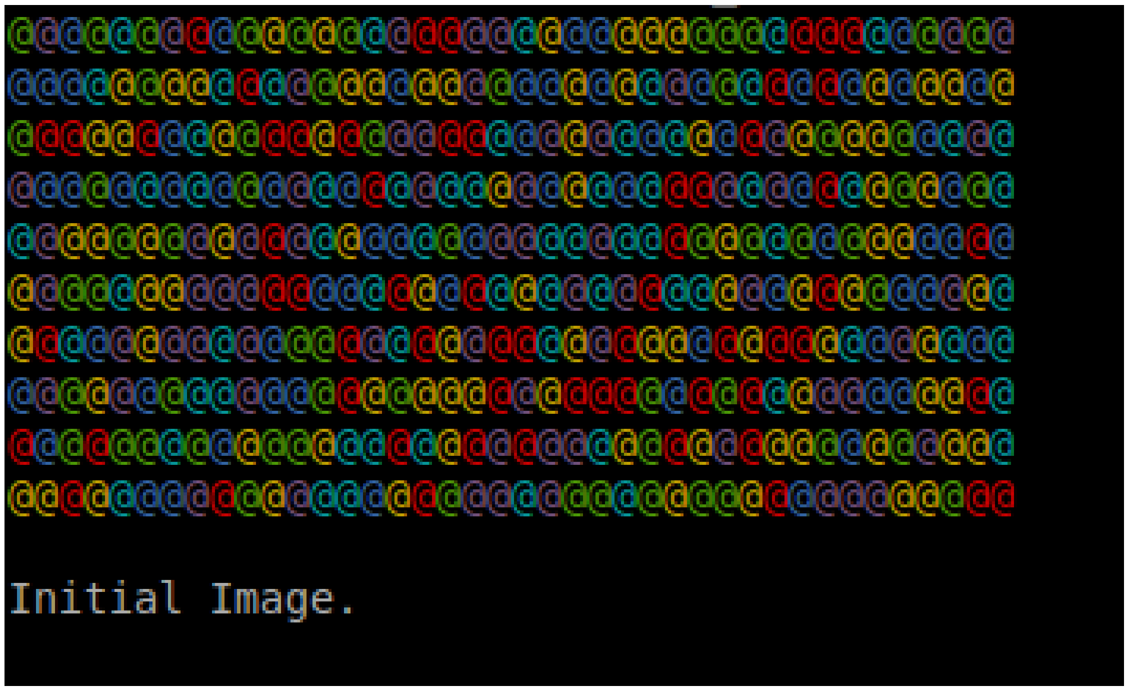


One operation per pixel

Part 1 – Image processing

- We represent the picture as a 2D matrix

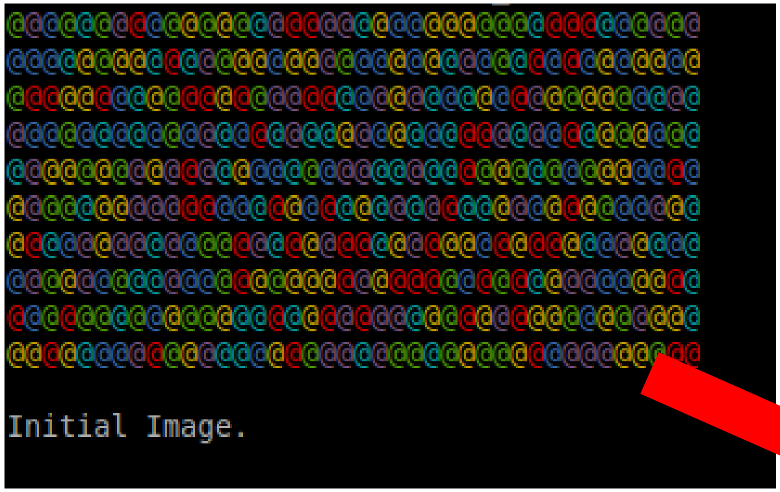




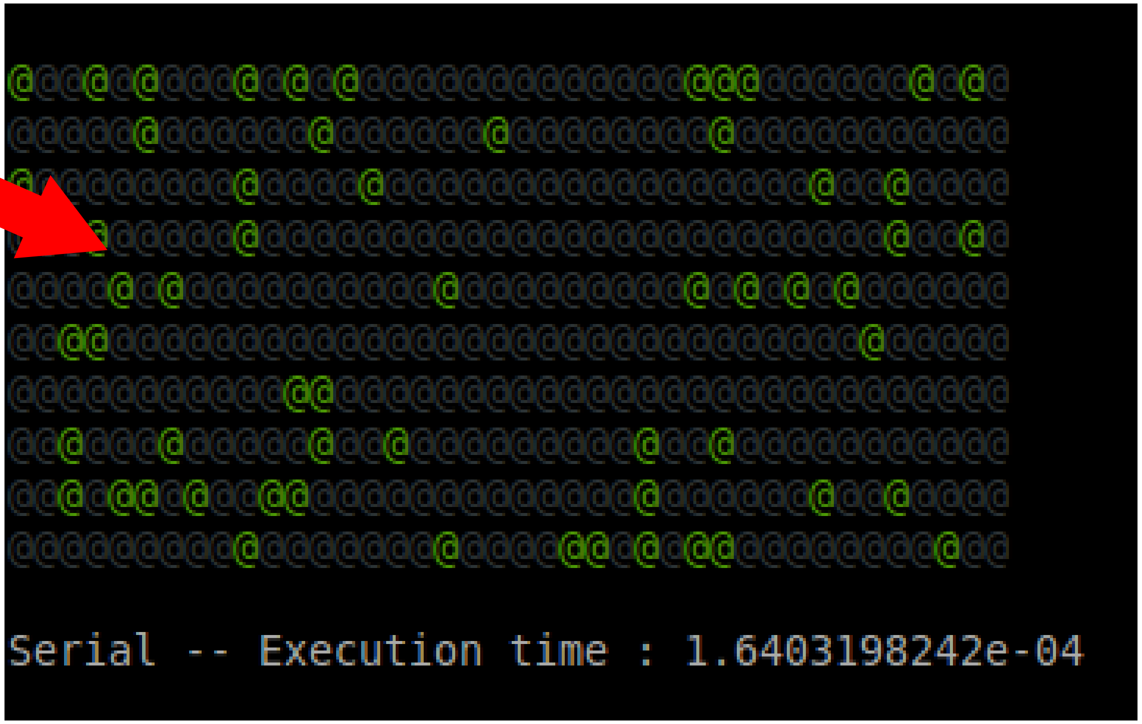
```
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@

Initial Image.
```

Part 1 – Image processing

- In a serial way, we can apply the operator per pixel

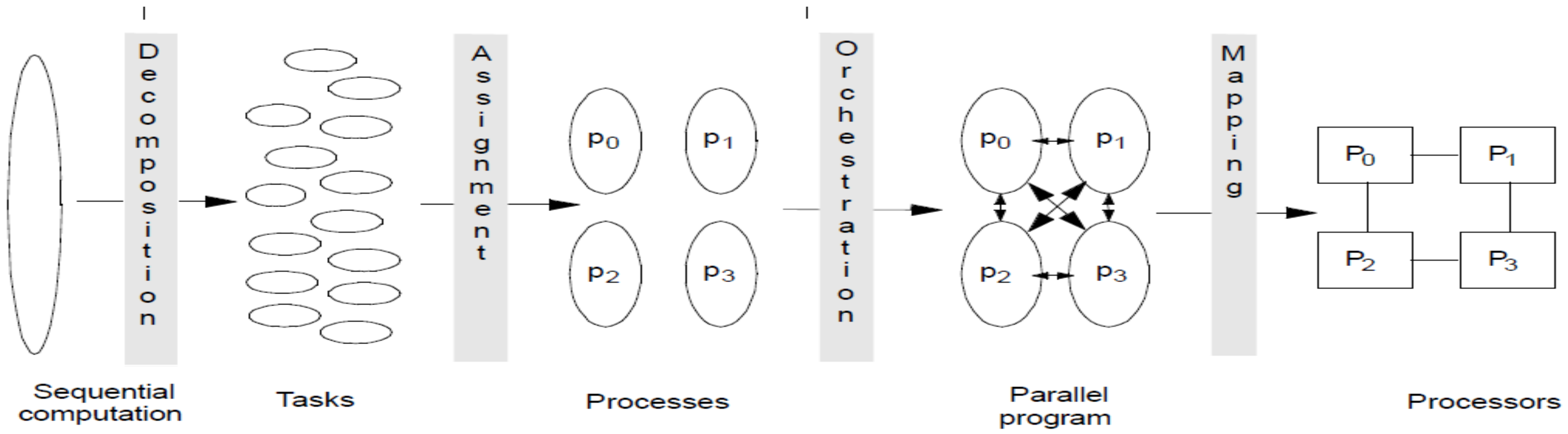


```
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@

Initial Image.
```



```
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@

Serial -- Execution time : 1.6403198242e-04
```

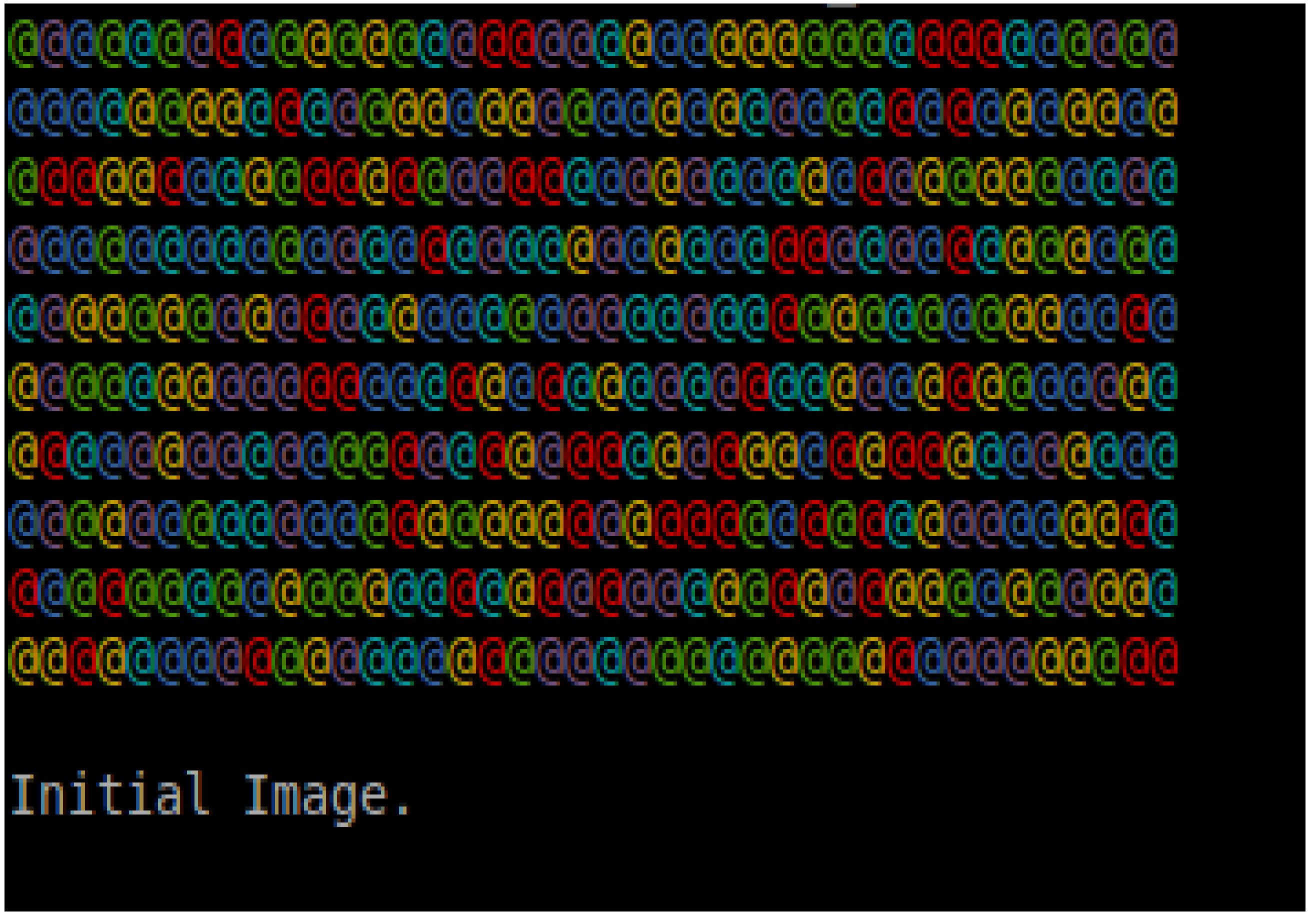
Introduction to Lab 1



1) Task to be performed – Image processing
2) **Parallelization scheme – Processes**
3) Synchronization scheme – Pipes
4) Report results

Part 2 – Parallelization scheme

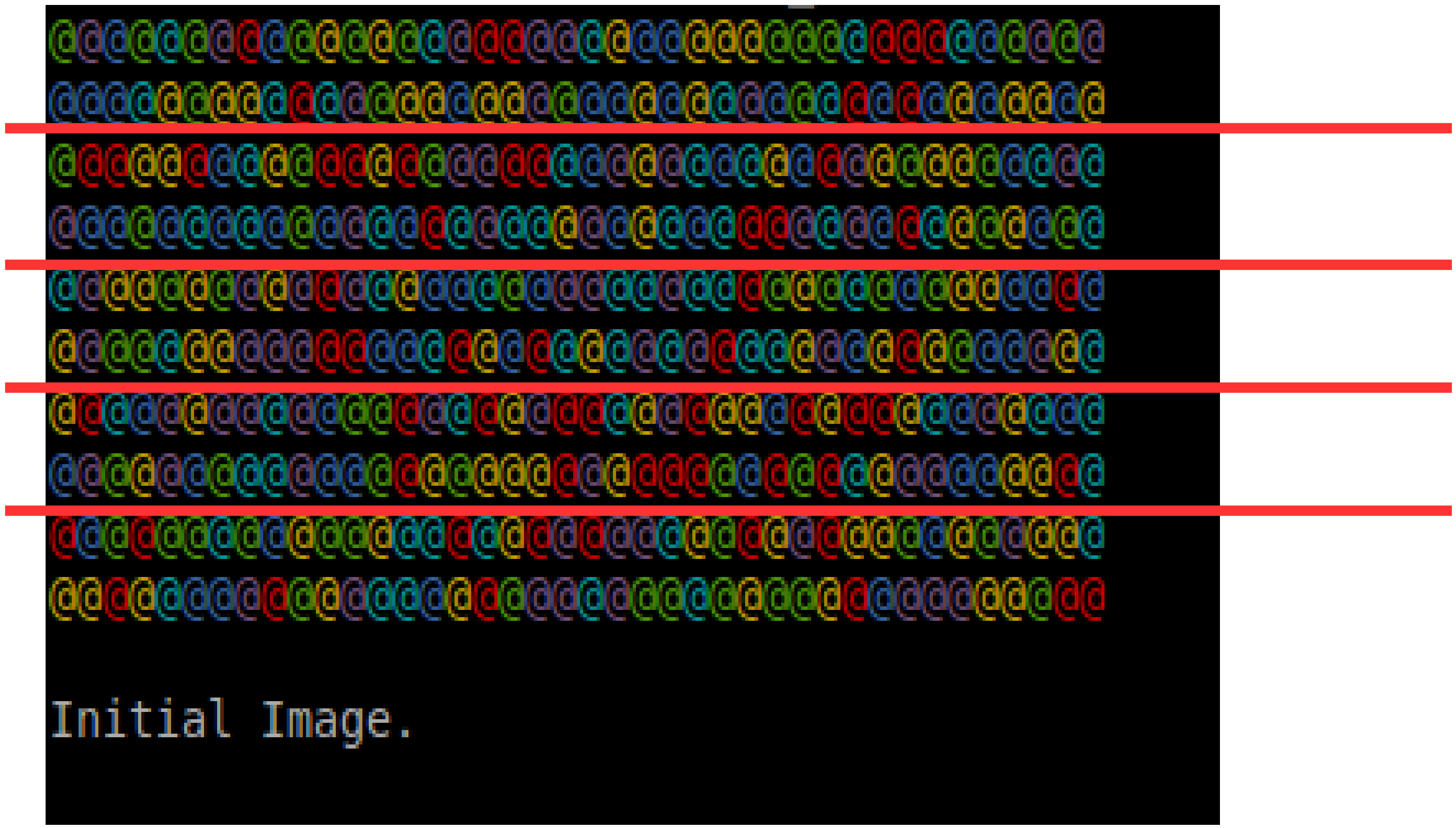
- We can devide the task per rows



```
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@

Initial Image.
```

Part 2 – Parallelization scheme

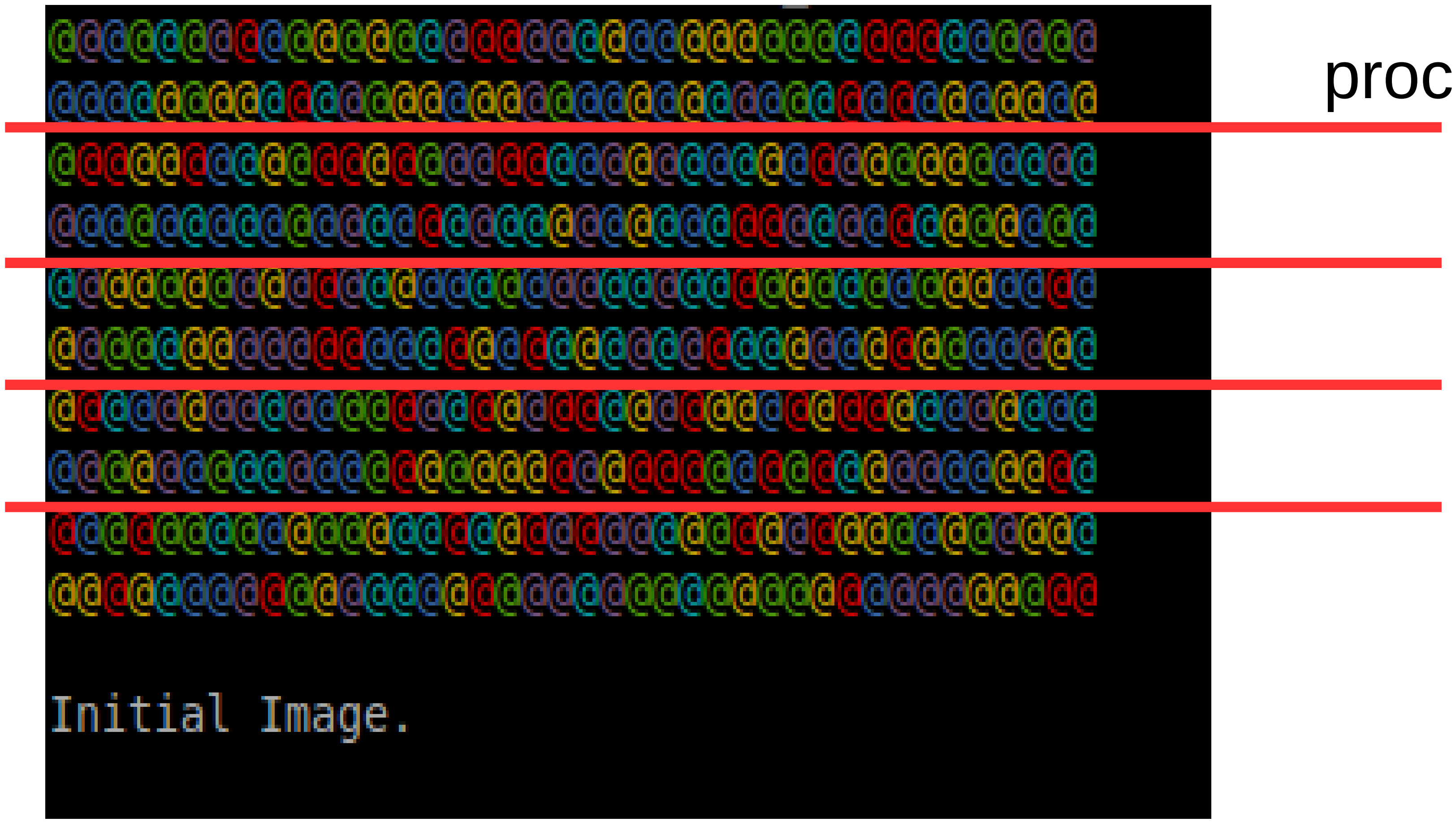
- We can devide the task per rows



```
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@

Initial Image.
```

Part 2 – Parallelization scheme

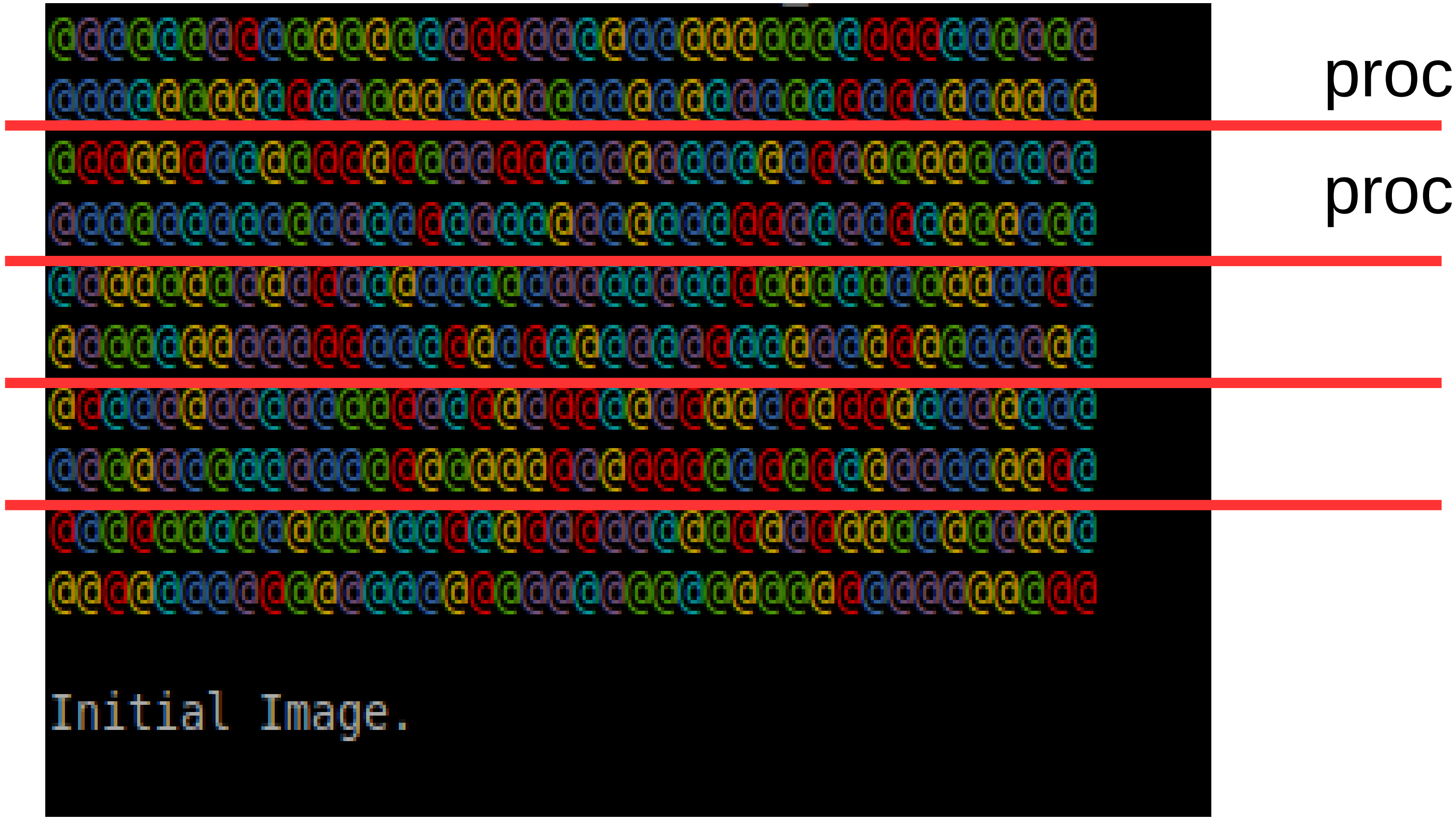
- We can devide the task per rows



```
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@

Initial Image.
```

Part 2 – Parallelization scheme

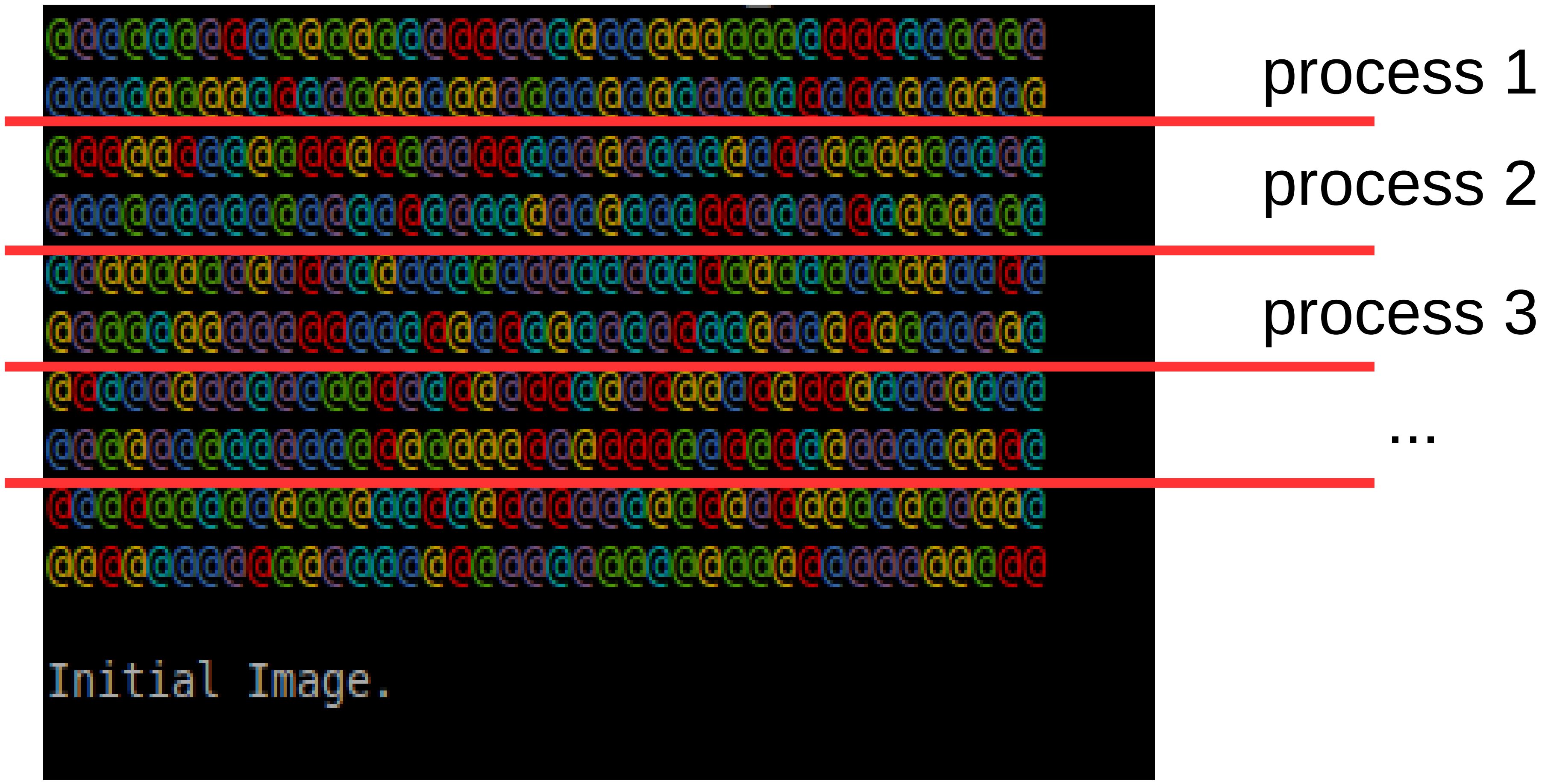
- We can devide the task per rows



```
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@

Initial Image.
```

Part 2 – Parallelization scheme

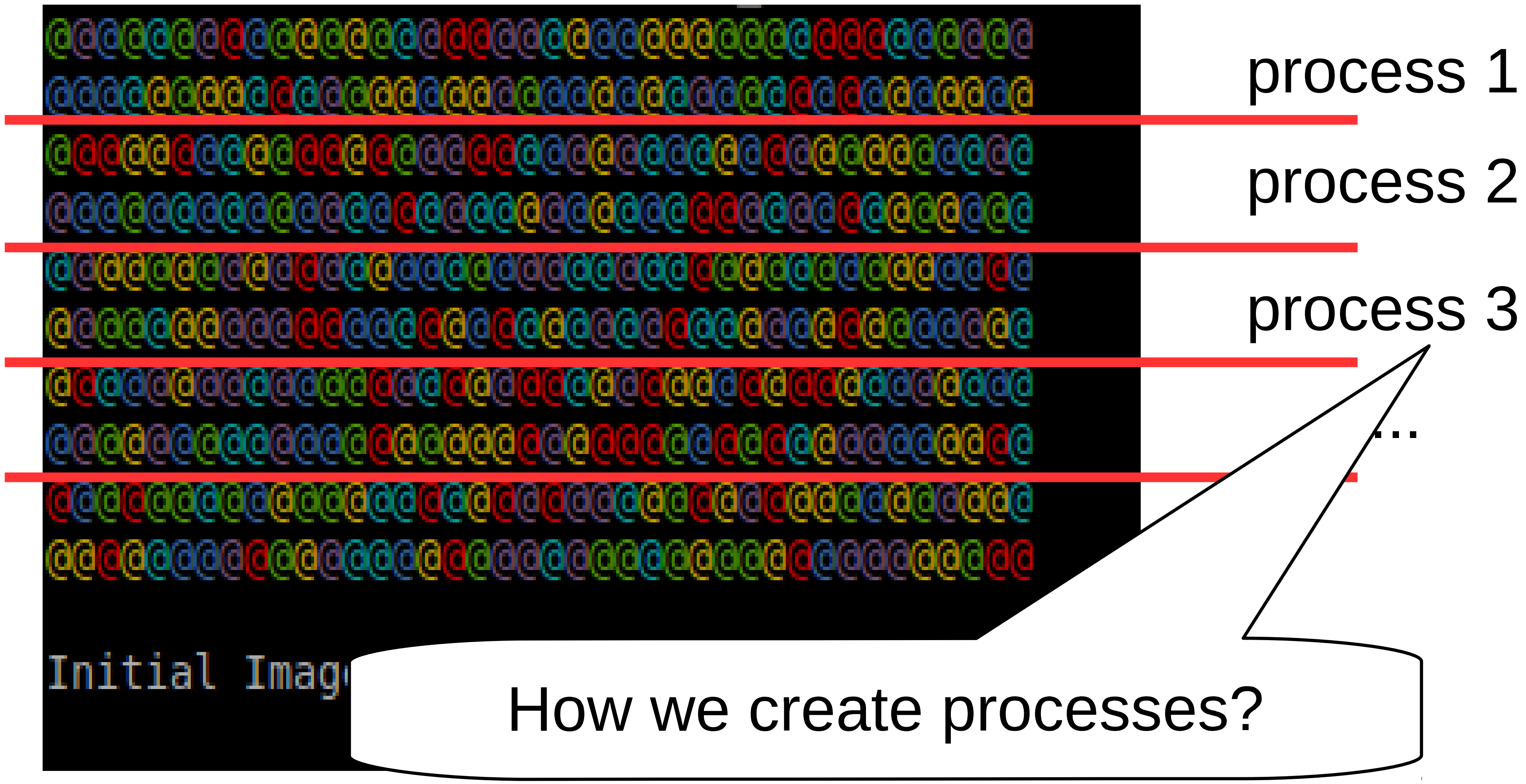
- We can devide the task per rows



```
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@

Initial Image.
```

Part 2 – Parallelization scheme

- We can devide the task per rows



```
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@

Initial Imag
```

How we create processes?

Part 2 - How to create a process : fork()

PID=832 ~ Parent

- **Program**

Same program →

PID=864 ~ Child

- **Program**

Parent

- **State**

Process ID (PID)

Process State

Architectural State
(Register file, PC etc)

Address space
(Stack etc)

File descriptors,
Environment vars etc
...

Child

- **State**

Process ID (PID) *NEW*

Process State *NEW*

Architectural State *NEW*
(Register file, PC etc)

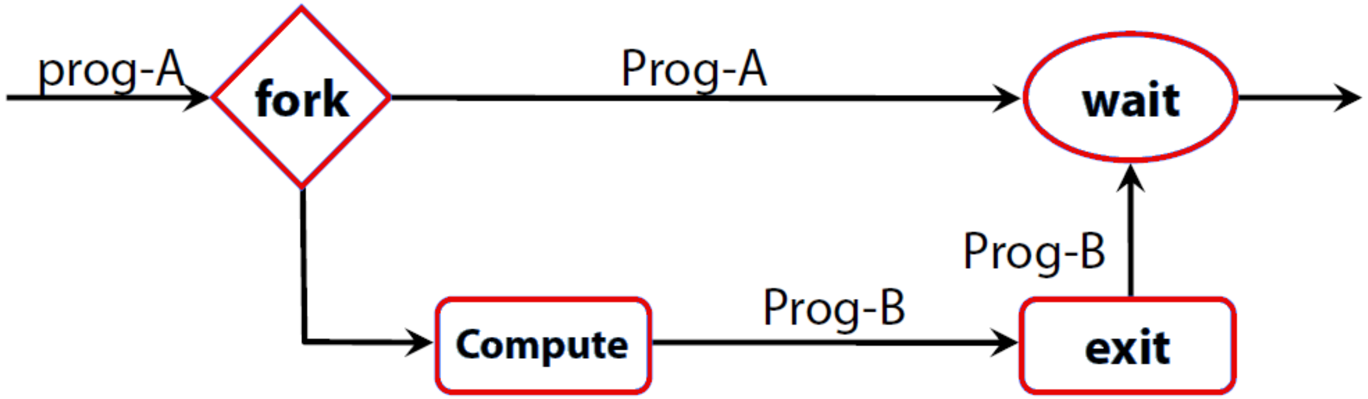
Address space *exact COPY*
(Stack etc)

File descriptors, *Inherited by child*
Environment vars etc
...

Part 2 - How to create a process : fork()

PID=832

```
pid_t p, mypid;
...
p = fork();
if (p < 0) {
      perror("fork");
      exit(1);
} else if (p == 0) {
      mypid = getpid();
      child();
} else {
      father();
}
```



p = ?   mypid = ?

How to create a process : fork()

PID=832

```
pid_t p, mypid;
...
p = fork();
if (p < 0) {
      perror(“fork”);
      exit(1);
} else if (p == 0) {
      mypid = getpid();
      child();
} else {
      father();
}
```

p = ?   mypid = ?

How to create a process : fork()

PID=832

```
pid_t p, mypid;
...
p = fork();
if (p < 0) {
      perror("fork");
      exit(1);
} else if (p == 0) {
      mypid = getpid();
      child();
} else {
      father();
}
```

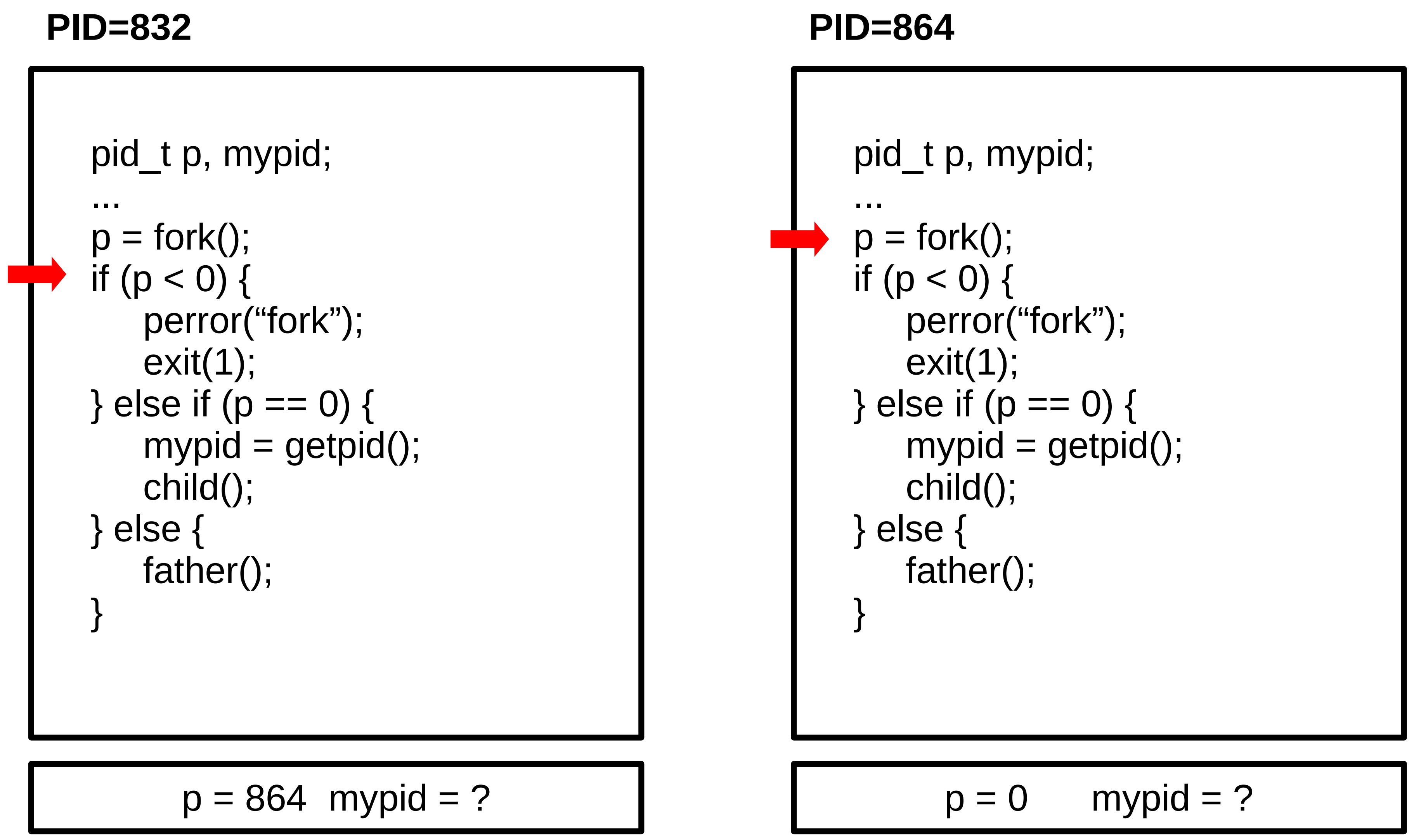
p = 864   mypid = ?

PID=864

```
pid_t p, mypid;
...
p = fork();
if (p < 0) {
      perror("fork");
      exit(1);
} else if (p == 0) {
      mypid = getpid();
      child();
} else {
      father();
}
```

p = 0   mypid = ?

How to create a process : fork()



PID=832

```
pid_t p, mypid;
...
p = fork();
if (p < 0) {
      perror("fork");
      exit(1);
} else if (p == 0) {
      mypid = getpid();
      child();
} else {
      father();
}
```

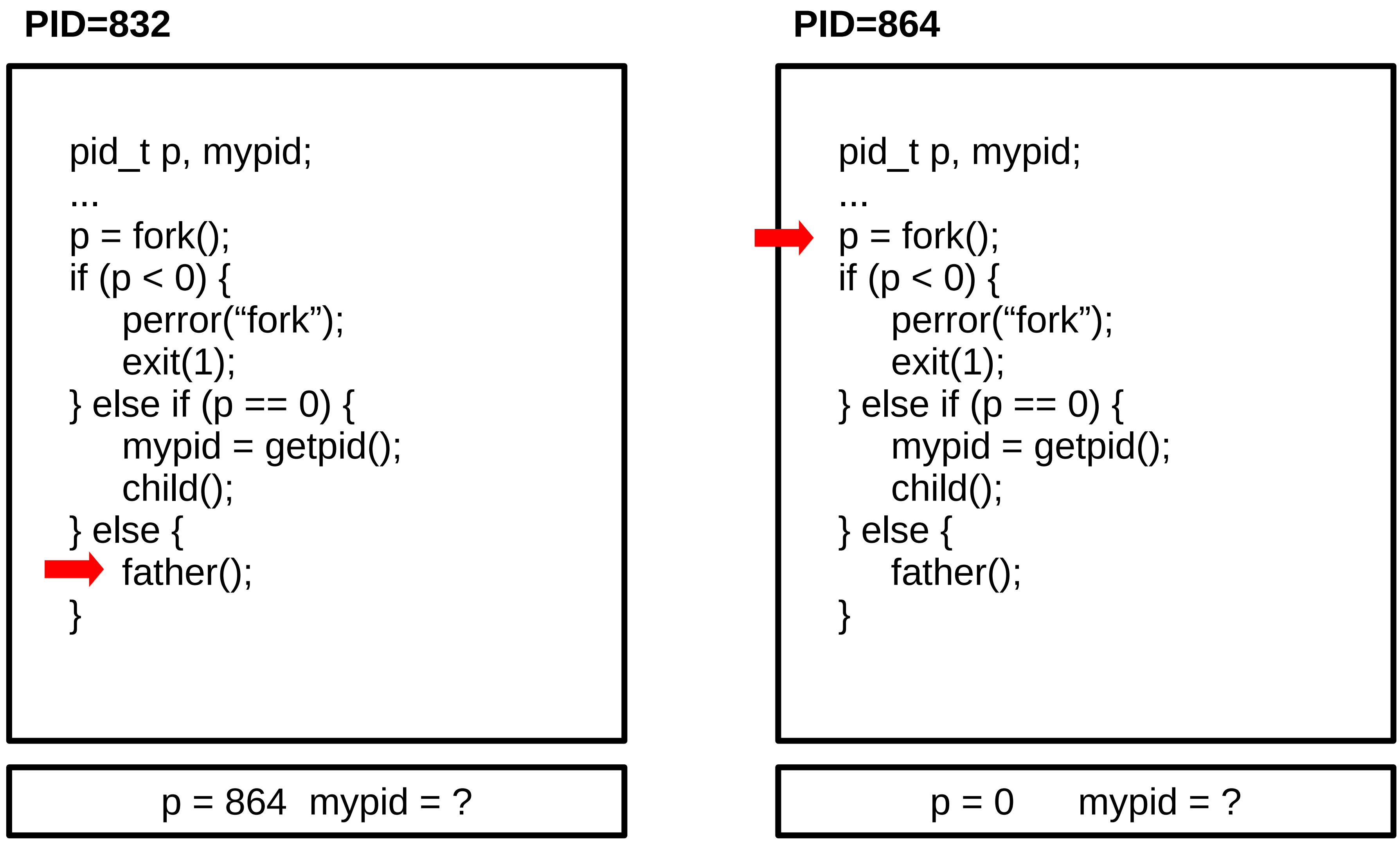
p = 864 mypid = ?

PID=864

```
pid_t p, mypid;
...
p = fork();
if (p < 0) {
      perror("fork");
      exit(1);
} else if (p == 0) {
      mypid = getpid();
      child();
} else {
      father();
}
```

p = 0 mypid = ?

How to create a process : fork()



PID=832

```
pid_t p, mypid;
...
p = fork();
if (p < 0) {
    perror(“fork”);
    exit(1);
} else if (p == 0) {
    mypid = getpid();
    child();
} else {
    father();
}
```

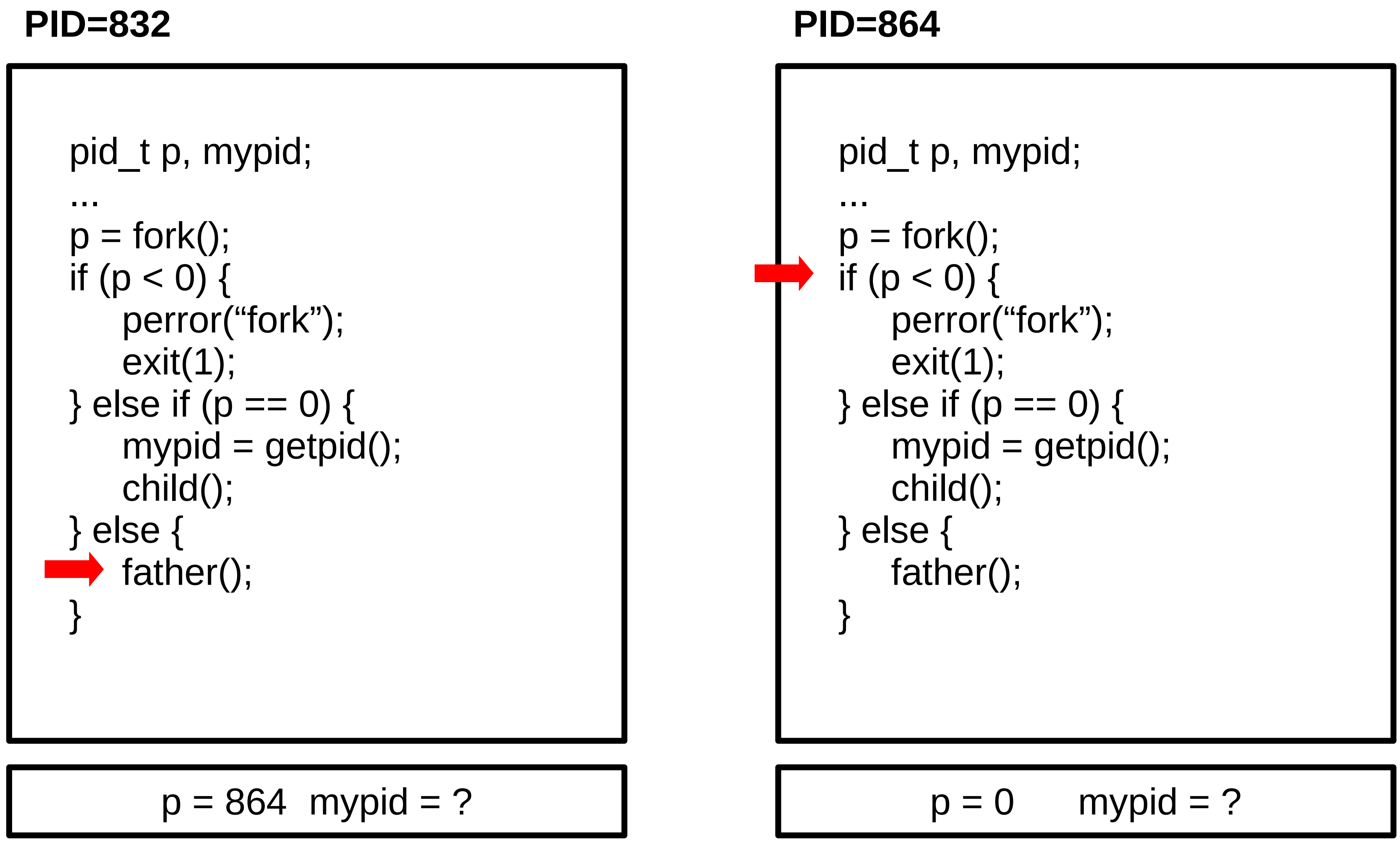
p = 864   mypid = ?

PID=864

```
pid_t p, mypid;
...
p = fork();
if (p < 0) {
    perror(“fork”);
    exit(1);
} else if (p == 0) {
    mypid = getpid();
    child();
} else {
    father();
}
```

p = 0   mypid = ?

How to create a process : fork()



PID=832

```
pid_t p, mypid;
...
p = fork();
if (p < 0) {
     perror(“fork”);
     exit(1);
} else if (p == 0) {
     mypid = getpid();
     child();
} else {
     father();
}
```

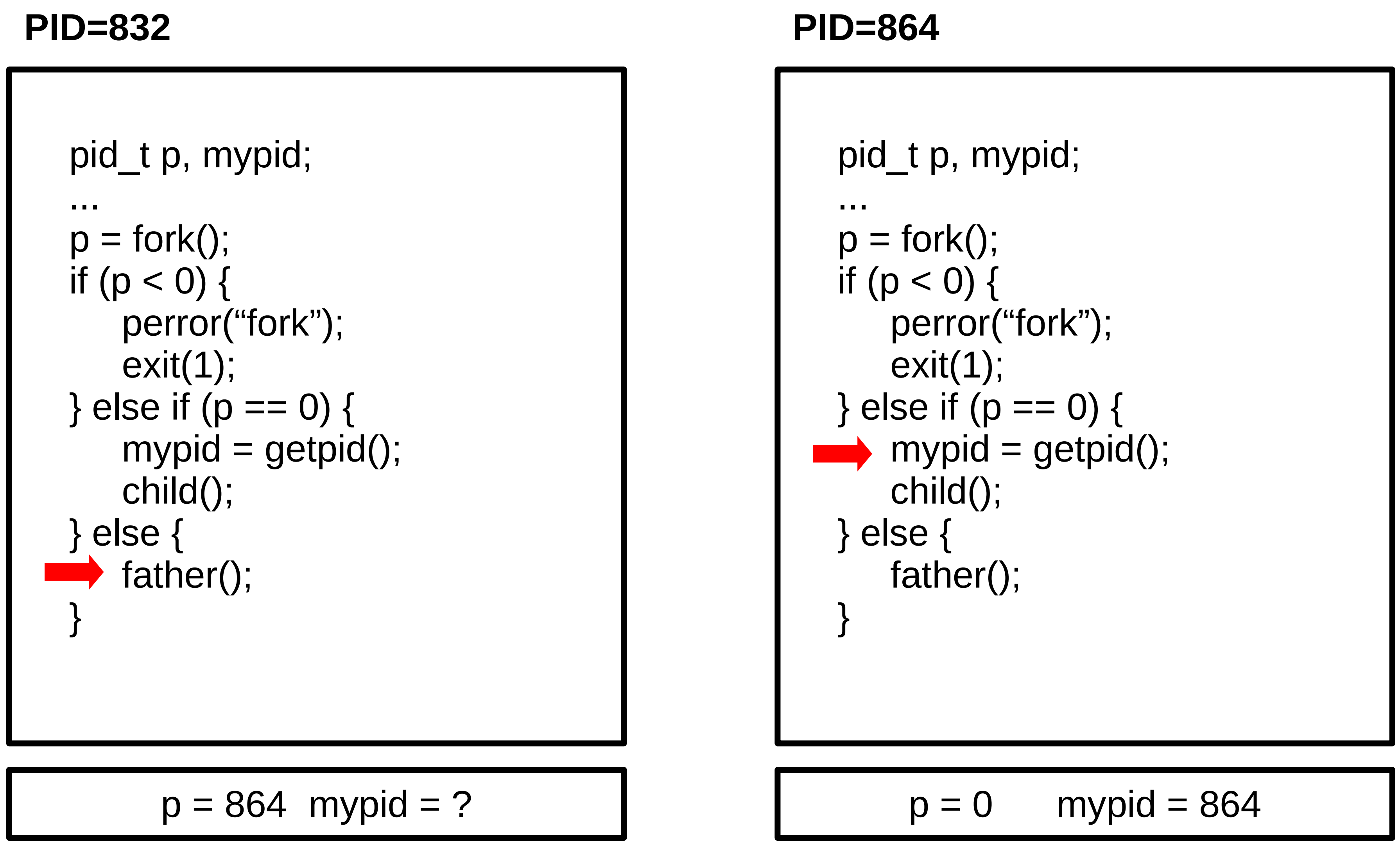
p = 864 mypid = ?

PID=864

```
pid_t p, mypid;
...
p = fork();
if (p < 0) {
     perror(“fork”);
     exit(1);
} else if (p == 0) {
     mypid = getpid();
     child();
} else {
     father();
}
```

p = 0   mypid = ?

How to create a process : fork()



PID=832

```
pid_t p, mypid;
...
p = fork();
if (p < 0) {
     perror("fork");
     exit(1);
} else if (p == 0) {
     mypid = getpid();
     child();
} else {
     father();
}
```

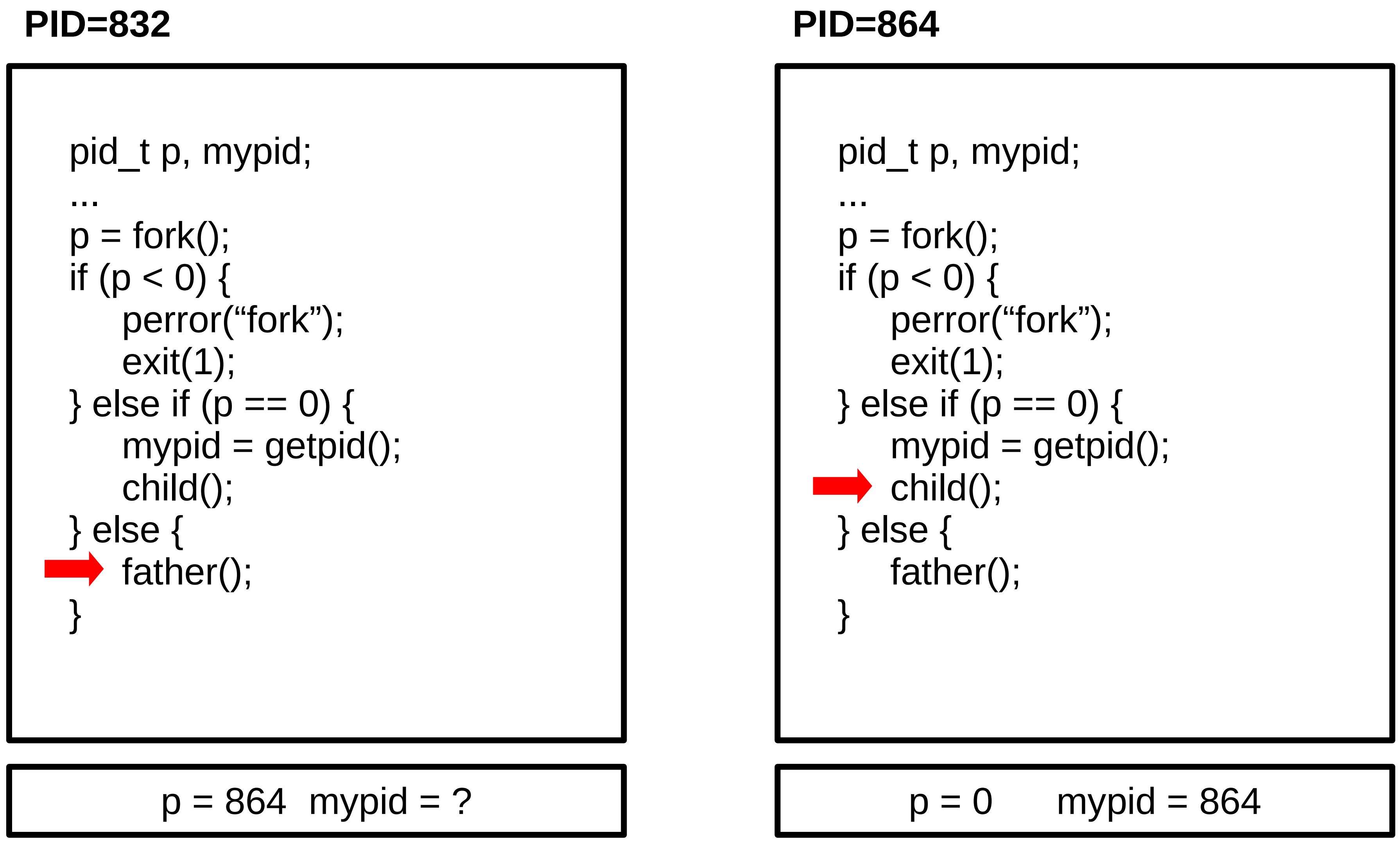
p = 864   mypid = ?

PID=864

```
pid_t p, mypid;
...
p = fork();
if (p < 0) {
     perror("fork");
     exit(1);
} else if (p == 0) {
     mypid = getpid();
     child();
} else {
     father();
}
```

p = 0   mypid = 864

How to create a process : fork()



PID=832

```
pid_t p, mypid;
...
p = fork();
if (p < 0) {
      perror("fork");
      exit(1);
} else if (p == 0) {
      mypid = getpid();
      child();
} else {
      father();
}
```

p = 864 mypid = ?

PID=864

```
pid_t p, mypid;
...
p = fork();
if (p < 0) {
      perror("fork");
      exit(1);
} else if (p == 0) {
      mypid = getpid();
      child();
} else {
      father();
}
```

p = 0   mypid = 864

How to create a process : fork()

PID=832

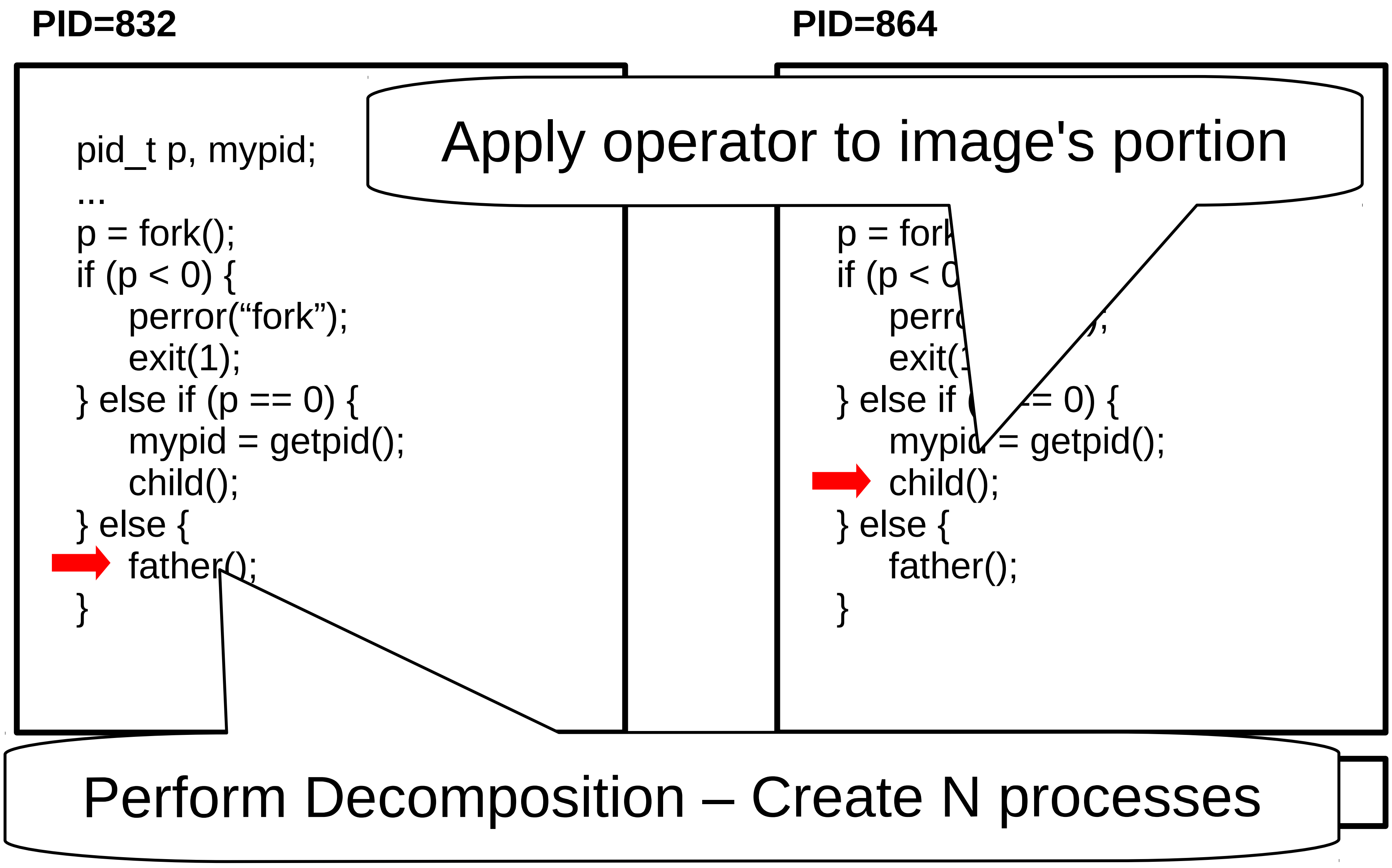
```
pid_t p, mypid;
...
p = fork();
if (p < 0) {
      perror("fork");
      exit(1);
} else if (p == 0) {
      mypid = getpid();
      child();
} else {
      father();
}
```

PID=864

```
pid_t p, mypid;
...
p = fork();
if (p < 0) {
      perror("fork");
      exit(1);
} else if (p == 0) {
      mypid = getpid();
      child();
} else {
      father();
}
```

Perform Decomposition – Create N processes

How to create a process : fork()



PID=832

```
pid_t p, mypid;
...
p = fork();
if (p < 0) {
     perror(“fork”);
     exit(1);
} else if (p == 0) {
     mypid = getpid();
     child();
} else {
     father();
}
```

PID=864

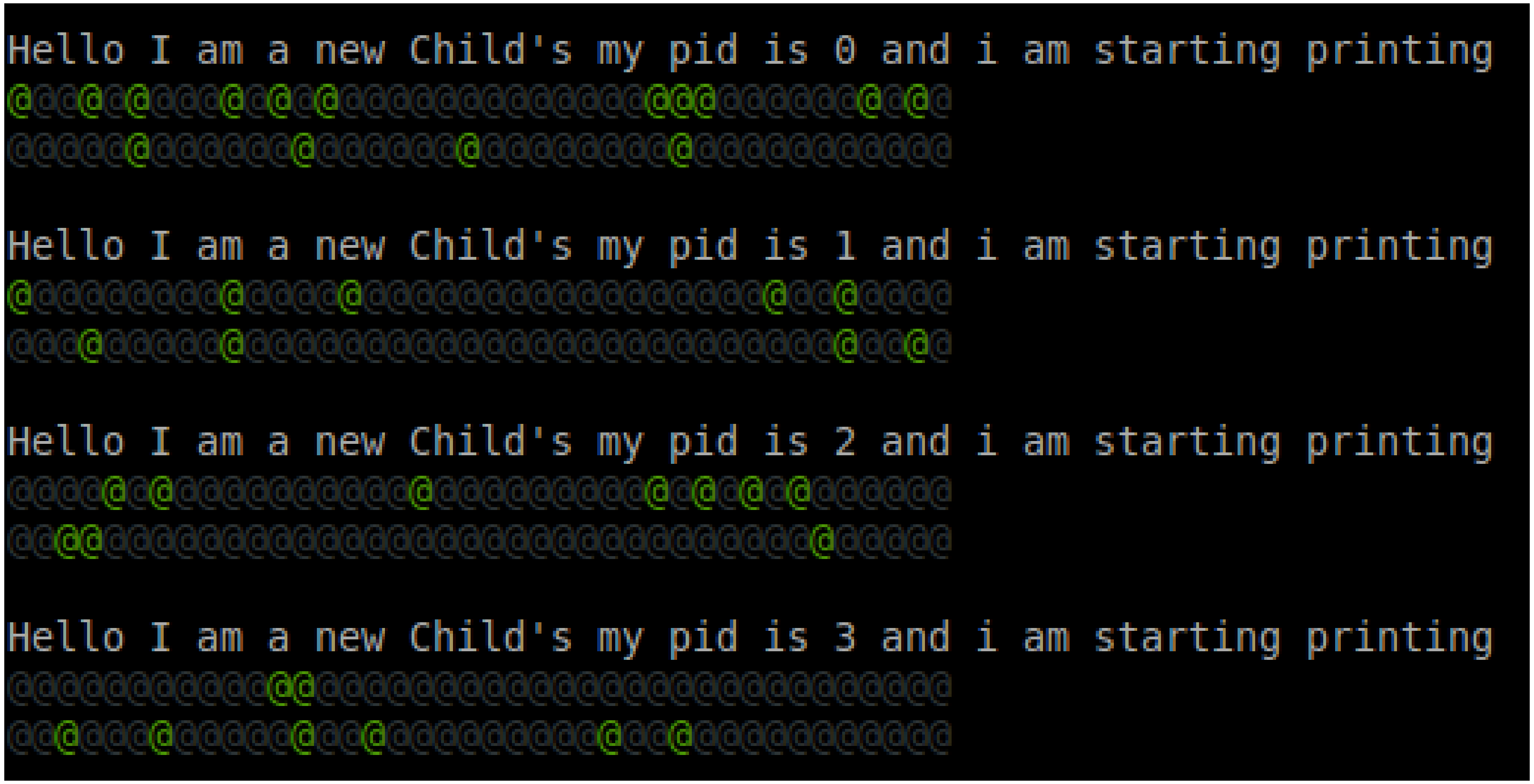
```
p = fork
if (p < 0
     perr           );
     exit(1
} else if (    == 0) {
     mypid = getpid();
     child();
} else {
     father();
}
```

Apply operator to image's portion

Perform Decomposition – Create N processes

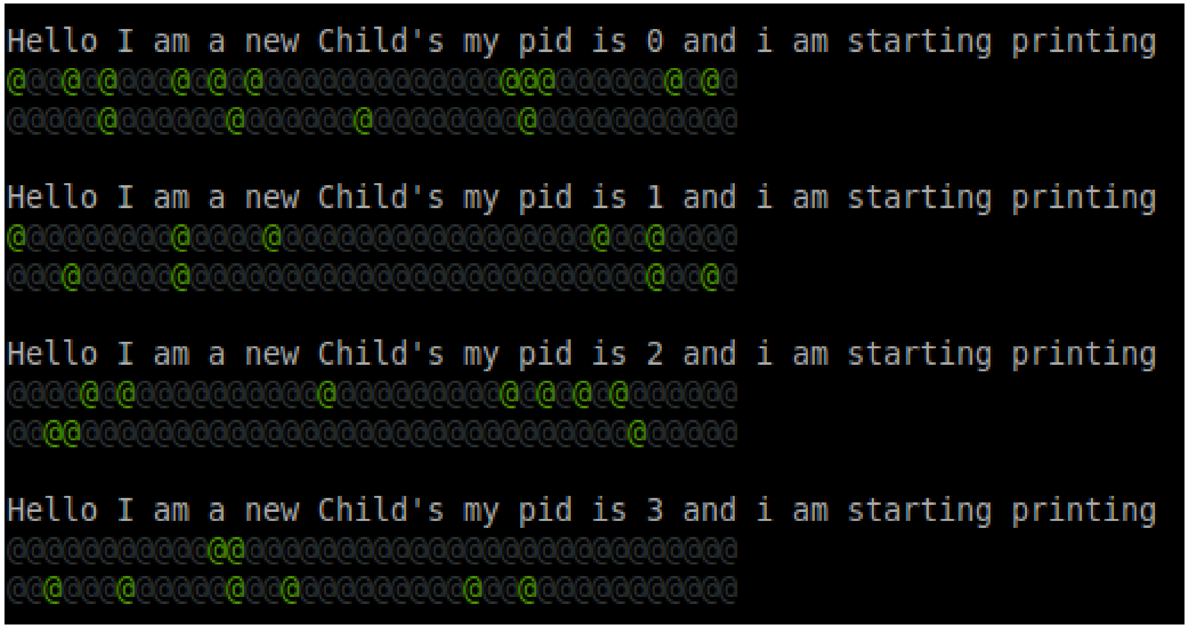
Part 2 – Parallelization scheme

- Let's print the results



```
Hello I am a new Child's my pid is 0 and i am starting printing
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@

Hello I am a new Child's my pid is 1 and i am starting printing
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@

Hello I am a new Child's my pid is 2 and i am starting printing
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@

Hello I am a new Child's my pid is 3 and i am starting printing
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
```

Part 2 – Parallelization scheme

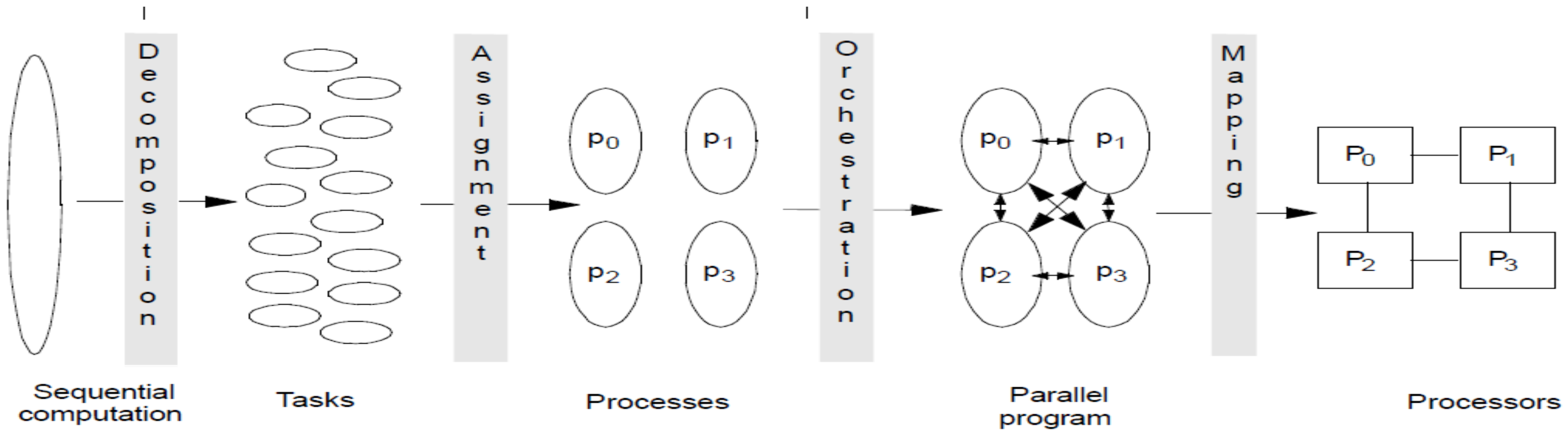


```
Hello I am a new Child's my pid is 0 and i am starting printing
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@

Hello I am a new Child's my pid is 1 and i am starting printing
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@

Hello I am a new Child's my pid is 2 and i am starting printing
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@

Hello I am a new Child's my pid is 3 and i am starting printing
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
```

By this point:

- Report the achieved speedup.
- How different number of processes affects performance?
- ...

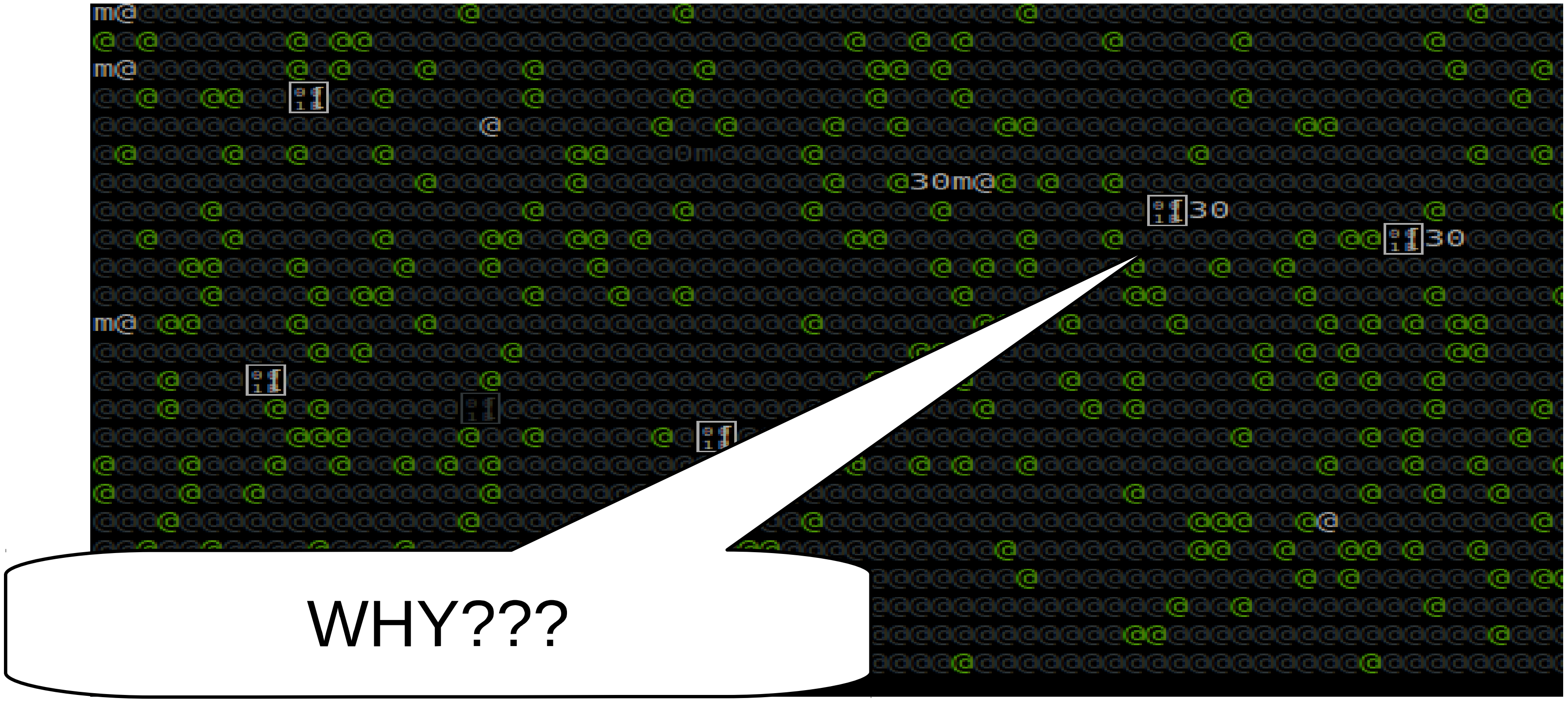
Introduction to Lab 1



1) Task to be performed – Image processing
2) Parallelization scheme – Processes
3) **Synchronization scheme – Pipes**
4) Report results

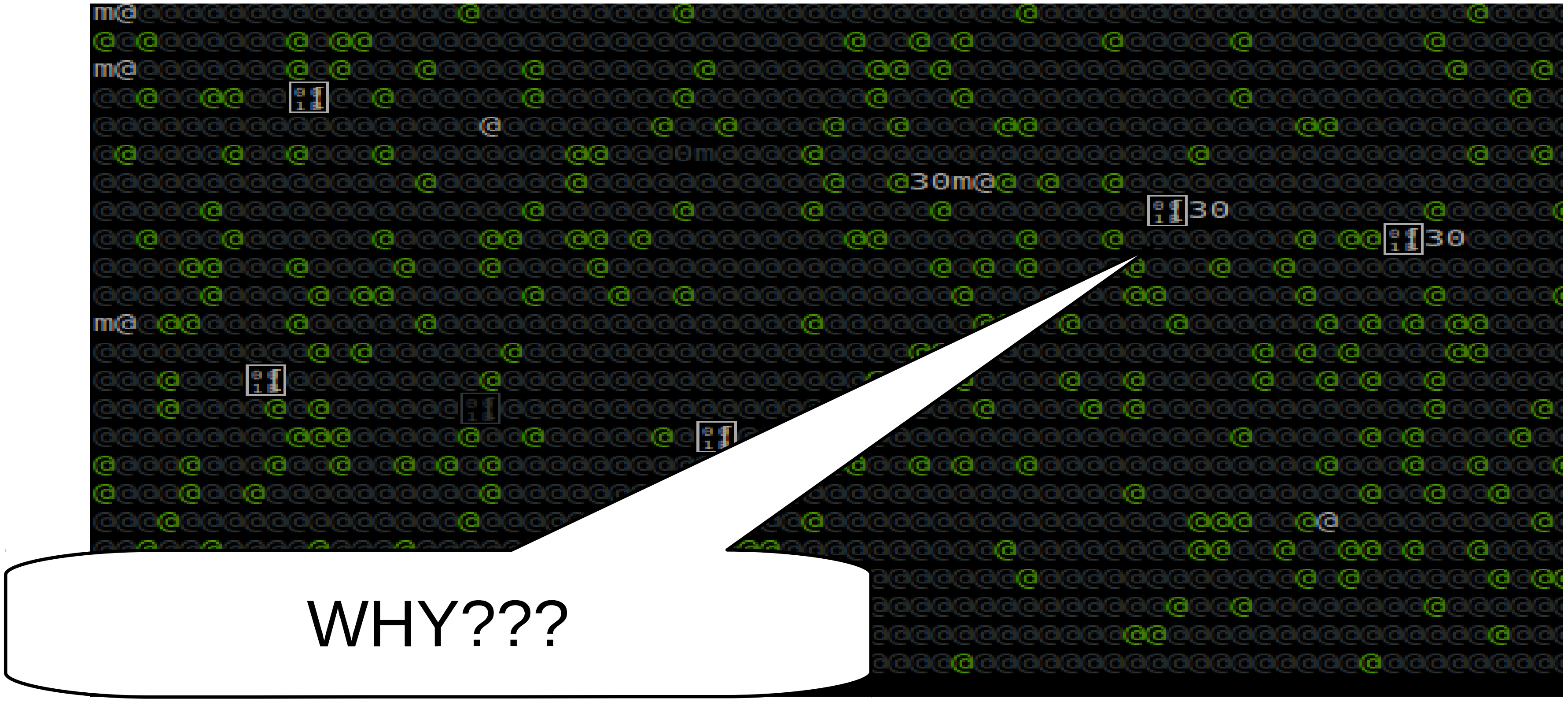
Part 3 – Synchronization scheme

- For bigger matrices, I have inconsistencies



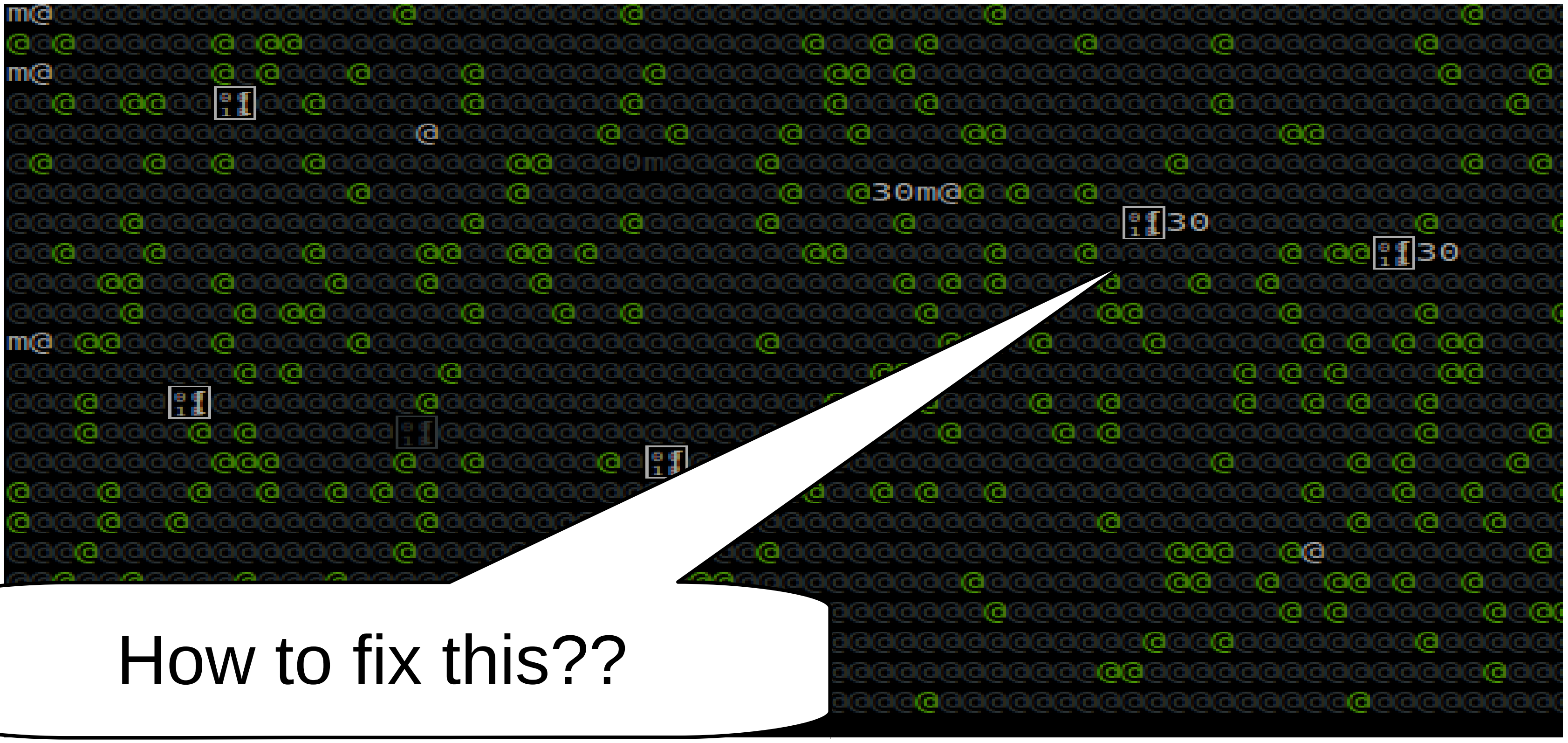
Part 3 – Synchronization scheme

- For bigger matrices, I have inconsistencies



More than one process tries to print at the same time..!!

Part 3 – Synchronization scheme

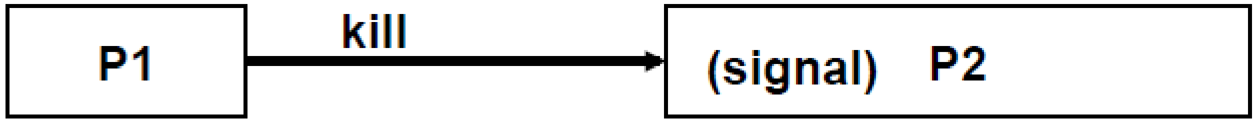


How to fix this??

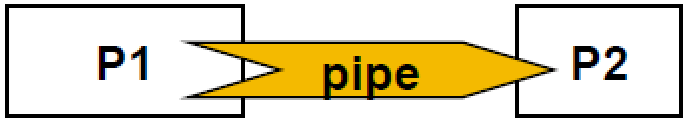
Inter-Process Communication (IPC)

IPC Mechanisms

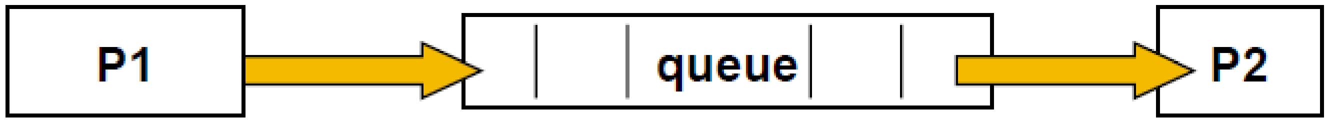
- Signals



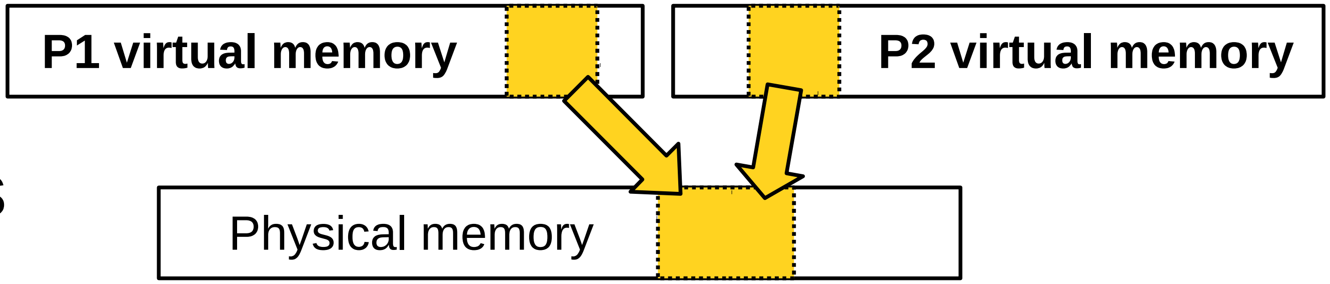
- Pipes



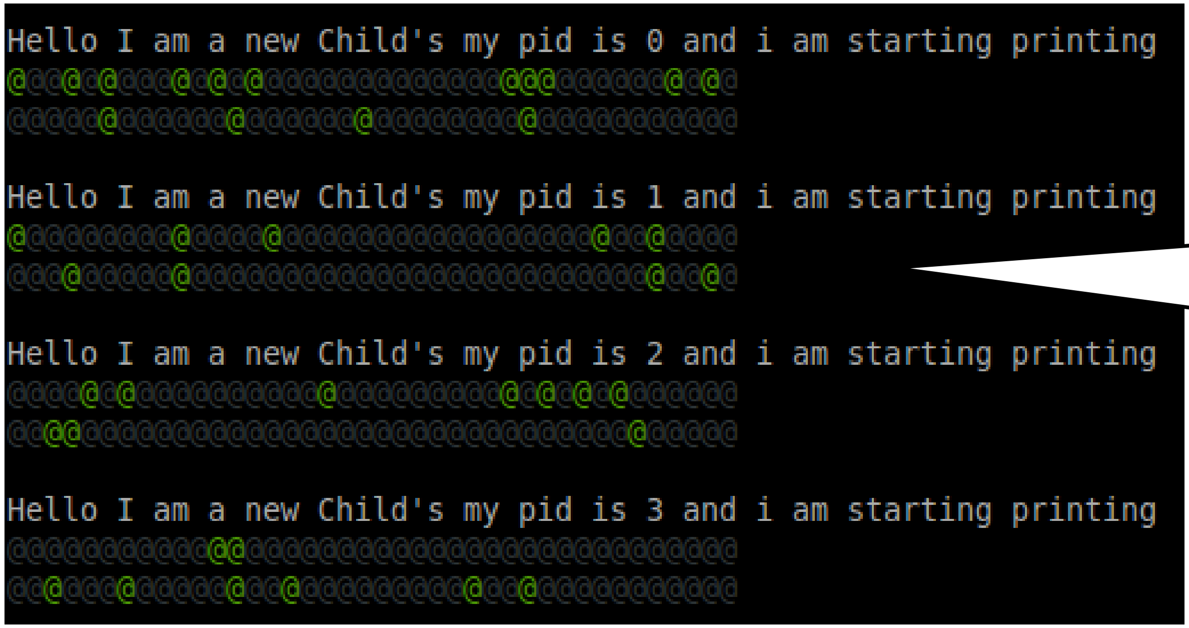
- POSIX messages



- Shared Memory Segments



Part 2 – Parallelization scheme



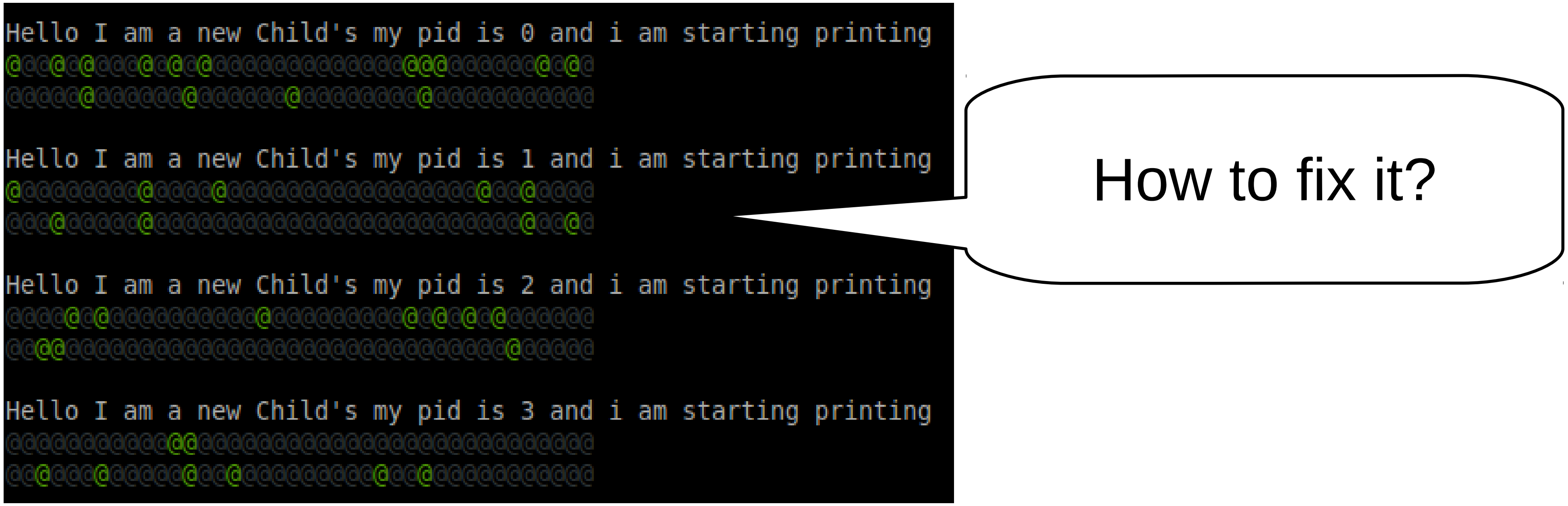
```
Hello I am a new Child's my pid is 0 and i am starting printing
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@

Hello I am a new Child's my pid is 1 and i am starting printing
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@

Hello I am a new Child's my pid is 2 and i am starting printing
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@

Hello I am a new Child's my pid is 3 and i am starting printing
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
```

How to fix it?

Inter-Process Communication (IPC):

- …
- …
- …
- …
- …

Part 2 – Parallelization scheme

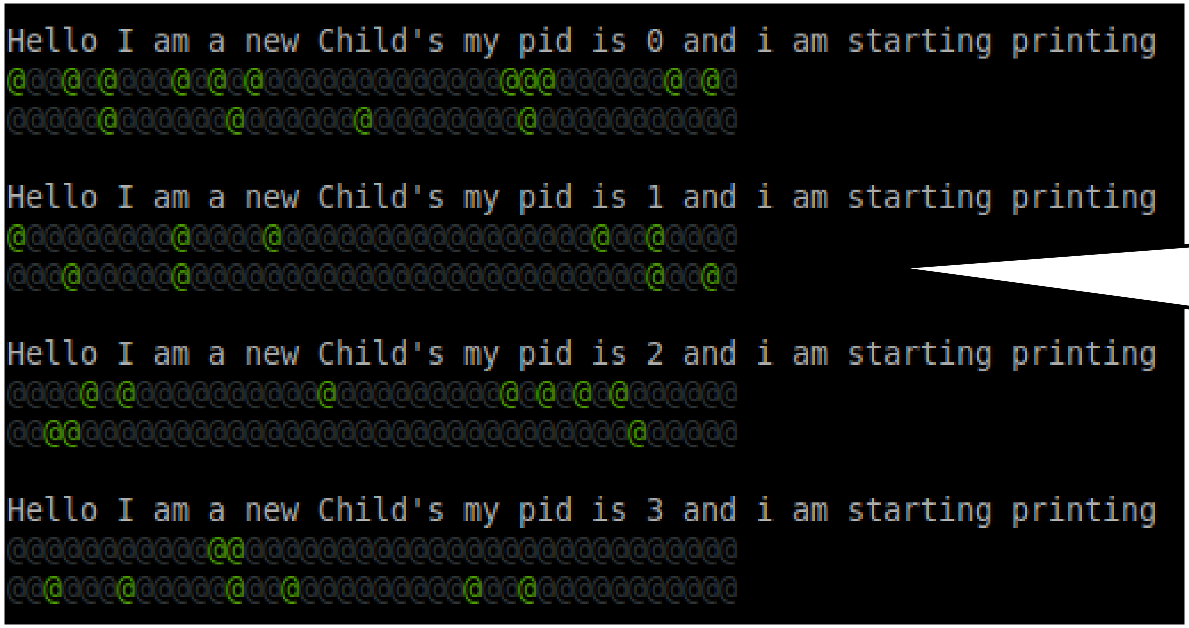


```
Hello I am a new Child's my pid is 0 and i am starting printing
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@

Hello I am a new Child's my pid is 1 and i am starting printing
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@

Hello I am a new Child's my pid is 2 and i am starting printing
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@

Hello I am a new Child's my pid is 3 and i am starting printing
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
```

Inter-Process Communication (IPC):

- Notify process 1 to print
- …
- …
- …
- …

Part 2 – Parallelization scheme



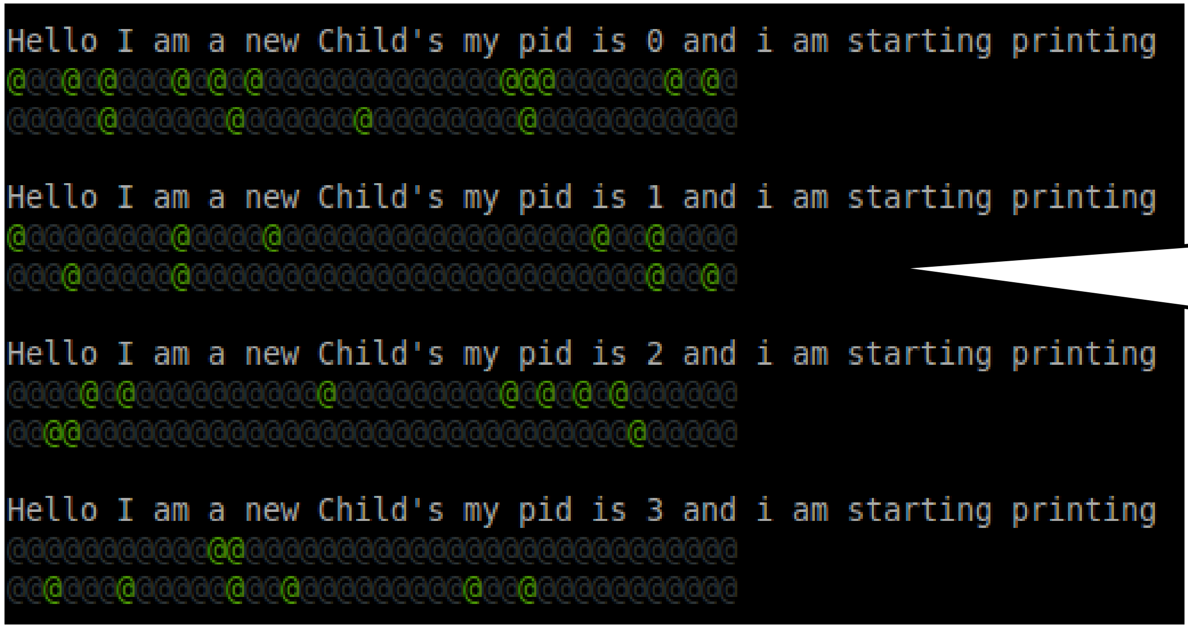
```
Hello I am a new Child's my pid is 0 and i am starting printing
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@

Hello I am a new Child's my pid is 1 and i am starting printing
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@

Hello I am a new Child's my pid is 2 and i am starting printing
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@

Hello I am a new Child's my pid is 3 and i am starting printing
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
```

How to fix it?

Inter-Process Communication (IPC):

- Notify process 1 to print
- Let process 1 to print
- ...
- ...
- ...

Part 2 – Parallelization scheme



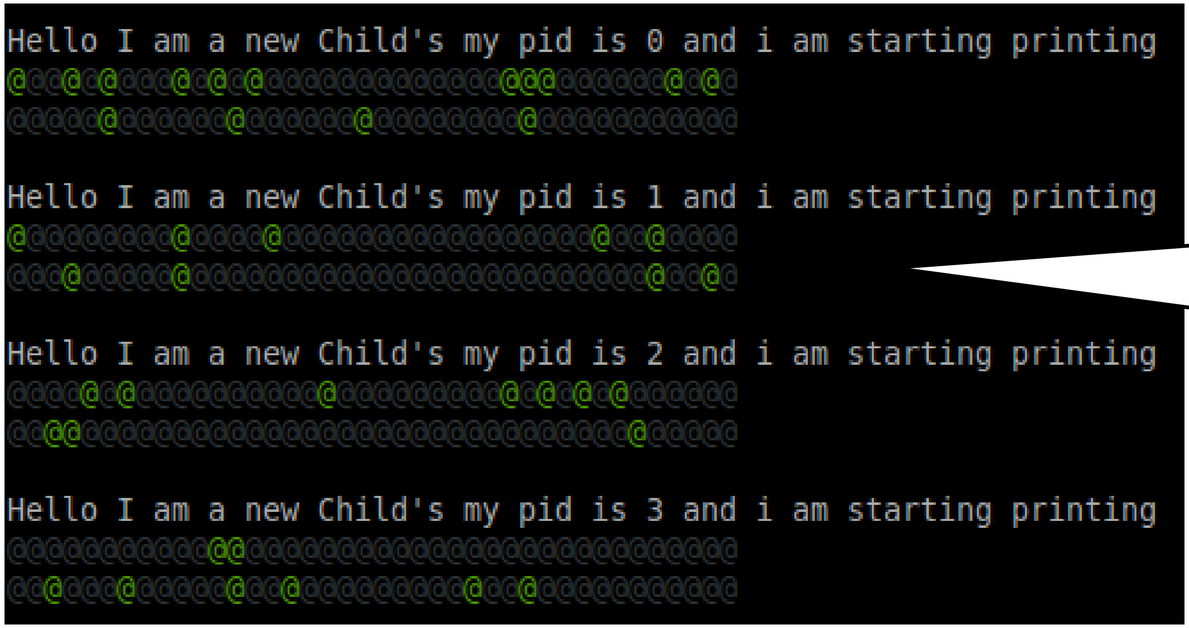
```
Hello I am a new Child's my pid is 0 and i am starting printing
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@

Hello I am a new Child's my pid is 1 and i am starting printing
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@

Hello I am a new Child's my pid is 2 and i am starting printing
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@

Hello I am a new Child's my pid is 3 and i am starting printing
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
```

How to fix it?

Inter-Process Communication (IPC):

- Notify process 1 to print
- Let process 1 to print
- Process 1 replies back that is done
- …
- …

Part 2 – Parallelization scheme



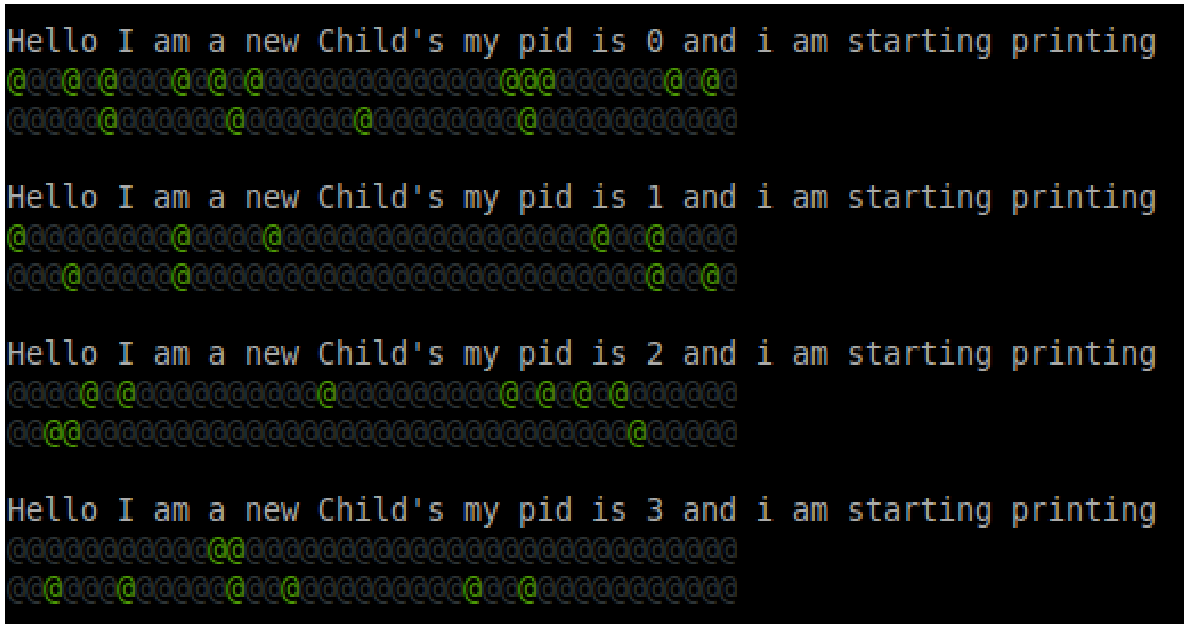
```
Hello I am a new Child's my pid is 0 and i am starting printing
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@

Hello I am a new Child's my pid is 1 and i am starting printing
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@

Hello I am a new Child's my pid is 2 and i am starting printing
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@

Hello I am a new Child's my pid is 3 and i am starting printing
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
```

How to fix it?

Inter-Process Communication (IPC):

- Notify process 1 to print
- Let process 1 to print
- Process 1 replies back that is done
- Process 2 is now notified
- ...

Part 2 – Parallelization scheme

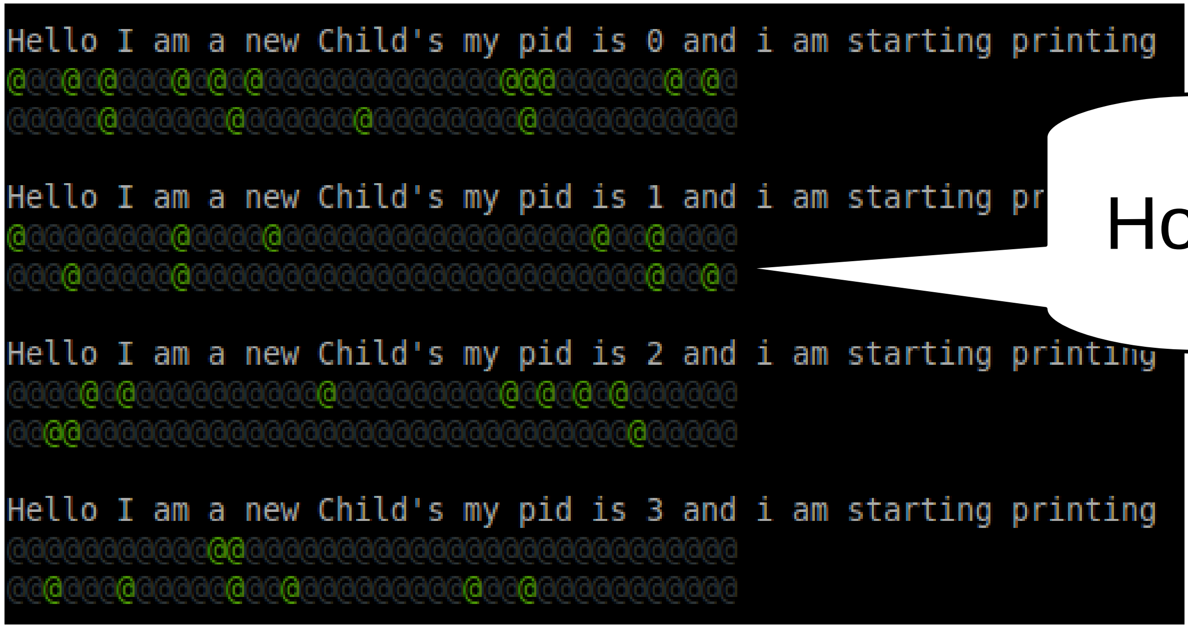


```
Hello I am a new Child's my pid is 0 and i am starting printing
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@

Hello I am a new Child's my pid is 1 and i am starting printing
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@

Hello I am a new Child's my pid is 2 and i am starting printing
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@

Hello I am a new Child's my pid is 3 and i am starting printing
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
```

Inter-Process Communication (IPC):

- Notify process 1 to print
- Let process 1 to print
- Process 1 replies back that is done
- Process 2 is now notified
- ...

Part 2 – Parallelization scheme

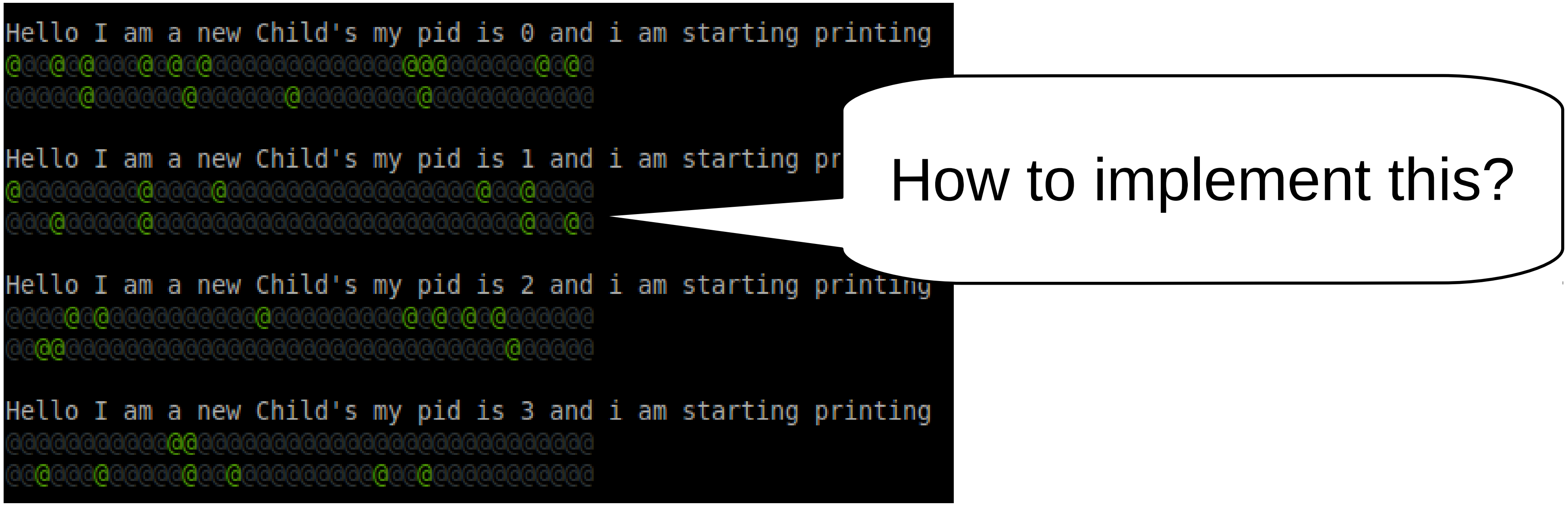


```
Hello I am a new Child's my pid is 0 and i am starting printing
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@

Hello I am a new Child's my pid is 1 and i am starting pr
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@

Hello I am a new Child's my pid is 2 and i am starting printing
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@

Hello I am a new Child's my pid is 3 and i am starting printing
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
```

How to implement this?

Inter-Process Communication (IPC):
- Notify process 1 to print
- Let process 1 to print
- Process 1 replies back that is done
- Process 2 is now notified
- ...

Part 2 – Parallelization scheme



```
Hello I am a new Child's my pid is 0 and i am starting printing
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@

Hello I am a new Child's my pid is 1 and i am starting pr
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@

Hello I am a new Child's my pid is 2 and i am starting printing
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@

Hello I am a new Child's my pid is 3 and i am starting printing
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
```

- We use pipes for messages to be communicated.
- Father will be responible for the synchronization.
- Children will be printing their portion of the image.

Part 3 – Pipes

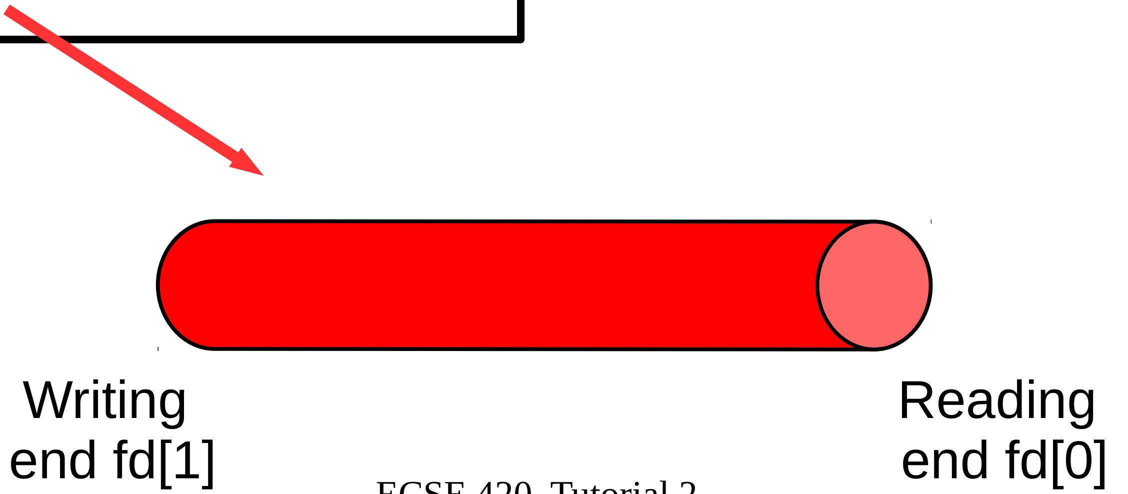
father

```
pid_t p, mypid;
int fd[2];
int n1, n2;
pipe(fd);
...
p = fork();
if (p == 0) {
  mypid = getpid();
  read(fd[0], &n2, sizeof(n2));
} else {
  write(fd[1], &n1, sizeof(n1));
}
```

Part 3 – Pipes

father

```
pid_t p, mypid;
int fd[2];
int n1, n2;
pipe(fd);
...
p = fork();
if (p == 0) {
  mypid = getpid();
  read(fd[0], &n2, sizeof(n2));
} else {
  write(fd[1], &n1, sizeof(n1));
}
```



Part 3 – Remember about fork

PID=832 ~ Parent | **PID=864 ~ Child**

- **Program** $\xrightarrow{\text{Same program}}$ - **Program**

- **State**

Process ID (PID)
Process State
Architectural State (Register file, PC etc)
Address space (Stack etc)
File descriptors, Environment vars etc ...

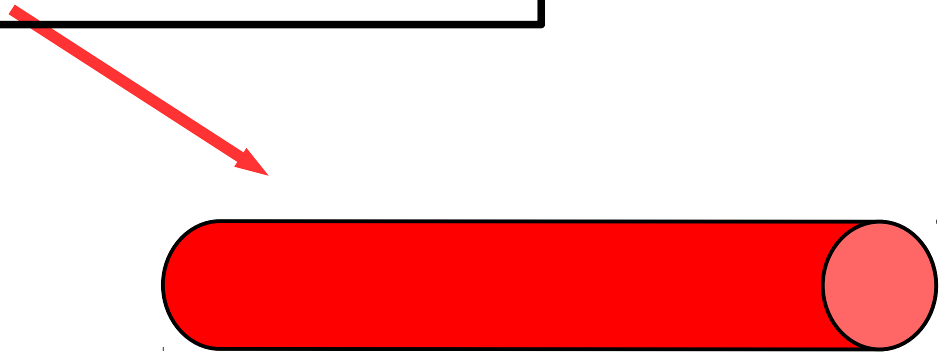
- **State**

Process ID (PID) *NEW*
Process State *NEW*
Architectural State *NEW* (Register file, PC etc)
Address space *exact COPY* (Stack etc)
File descriptors, *Inherited by child* Environment vars, Pipes etc ...

Part 3 – Pipes

father

```
pid_t p, mypid;
int fd[2];
int n1, n2;
pipe(fd);
...
p = fork();
if (p == 0) {
  mypid = getpid();
  read(fd[0], &n2, sizeof(n2));
} else {
  write(fd[1], &n1, sizeof(n1));
}
```



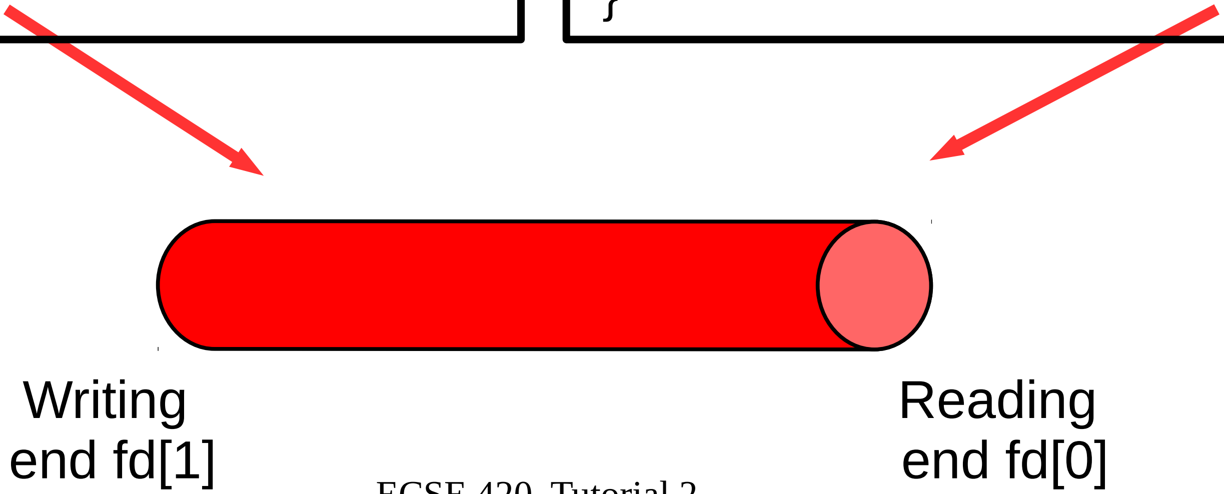
Part 3 – Pipes

father

```
pid_t p, mypid;
int fd[2];
int n1, n2;
pipe(fd);
...
p = fork();
if (p == 0) {
  mypid = getpid();
  read(fd[0], &n2, sizeof(n2));
} else {
  write(fd[1], &n1, sizeof(n1));
}
```

child

```
pid_t p, mypid;
int fd[2];
int n1, n2;
pipe(fd);
...
p = fork();
if (p == 0) {
  mypid = getpid();
  read(fd[0], &n2, sizeof(n2));
} else {
  write(fd[1], &n1, sizeof(n1));
}
```



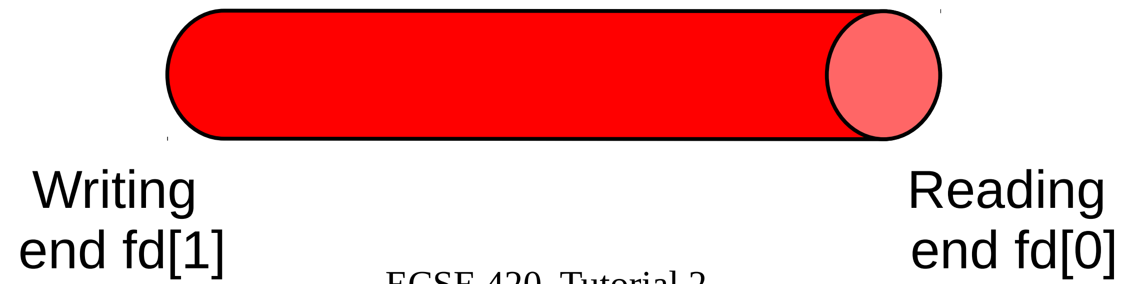
Part 3 – Communication scheme

father

```
pid_t p, mypid;
int fd[2];
int n1, n2;
pipe(fd);
...
p = fork();
if (p == 0) {
  mypid = getpid();
  read(fd[0], &n2, sizeof(n2));
} else {
  write(fd[1], &n1, sizeof(n1));
}
```

child

```
pid_t p, mypid;
int fd[2];
int n1, n2;
pipe(fd);
...
p = fork();
if (p == 0) {
  mypid = getpid();
  read(fd[0], &n2, sizeof(n2));
} else {
  write(fd[1], &n1, sizeof(n1));
}
```



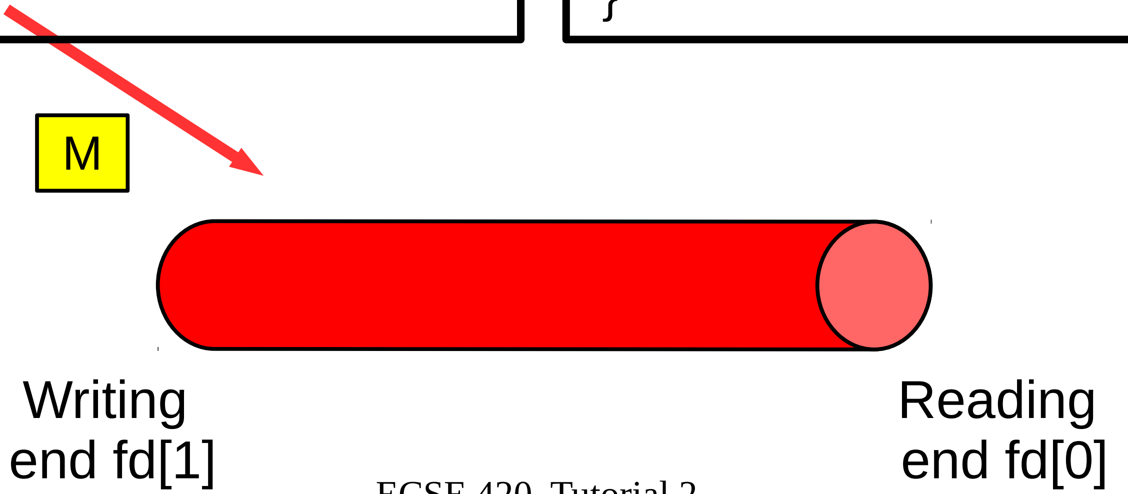
Part 3 – Communication scheme

father

```
pid_t p, mypid;
int fd[2];
int n1, n2;
pipe(fd);
...
p = fork();
if (p == 0) {
  mypid = getpid();
  read(fd[0], &n2, sizeof(n2));
} else {
  write(fd[1], &n1, sizeof(n1));
}
```

child

```
pid_t p, mypid;
int fd[2];
int n1, n2;
pipe(fd);
...
p = fork();
if (p == 0) {
  mypid = getpid();
  read(fd[0], &n2, sizeof(n2));
} else {
  write(fd[1], &n1, sizeof(n1));
}
```



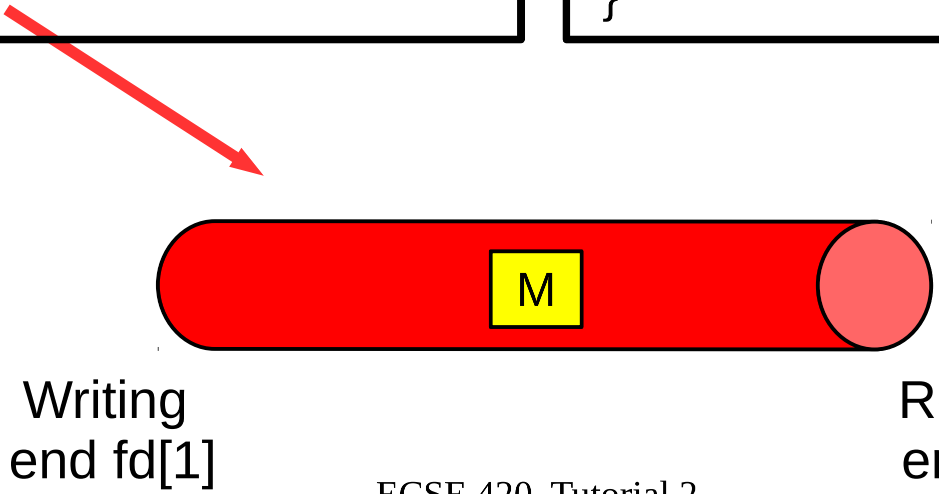
Part 3 – Communication scheme

father

```
pid_t p, mypid;
int fd[2];
int n1, n2;
pipe(fd);
...
p = fork();
if (p == 0) {
  mypid = getpid();
  read(fd[0], &n2, sizeof(n2));
} else {
  write(fd[1], &n1, sizeof(n1));
}
```

child

```
pid_t p, mypid;
int fd[2];
int n1, n2;
pipe(fd);
...
p = fork();
if (p == 0) {
  mypid = getpid();
  read(fd[0], &n2, sizeof(n2));
} else {
  write(fd[1], &n1, sizeof(n1));
}
```



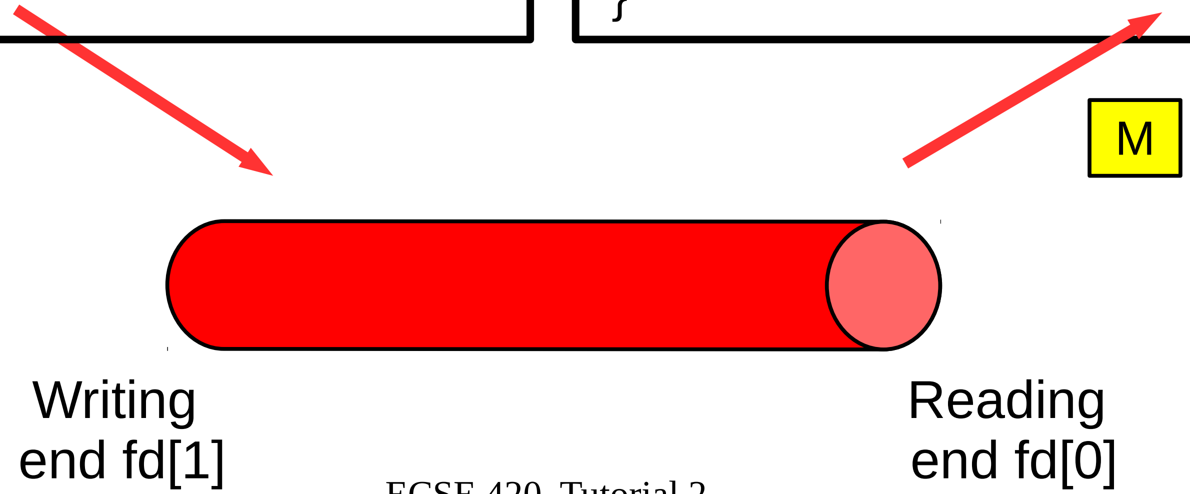
Part 3 – Communication scheme

father

```
pid_t p, mypid;
int fd[2];
int n1, n2;
pipe(fd);
...
p = fork();
if (p == 0) {
  mypid = getpid();
  read(fd[0], &n2, sizeof(n2));
} else {
  write(fd[1], &n1, sizeof(n1));
}
```

child

```
pid_t p, mypid;
int fd[2];
int n1, n2;
pipe(fd);
...
p = fork();
if (p == 0) {
  mypid = getpid();
  read(fd[0], &n2, sizeof(n2));
} else {
  write(fd[1], &n1, sizeof(n1));
}
```



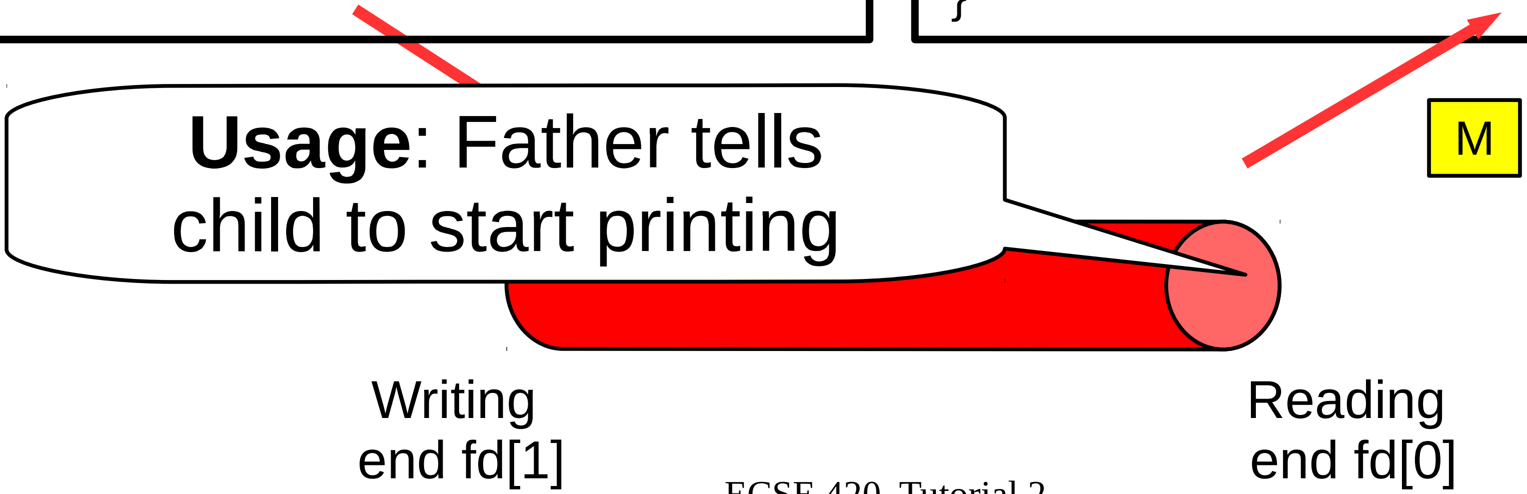
Part 3 – Communication scheme

father

```
pid_t p, mypid;
int fd[2];
int n1, n2;
pipe(fd);
...
p = fork();
if (p == 0) {
  mypid = getpid();
  read(fd[0], &n2, sizeof(n2));
} else {
  write(fd[1], &n1, sizeof(n1));
}
```

child

```
pid_t p, mypid;
int fd[2];
int n1, n2;
pipe(fd);
...
p = fork();
if (p == 0) {
  mypid = getpid();
  read(fd[0], &n2, sizeof(n2));
} else {
  write(fd[1], &n1, sizeof(n1));
}
```



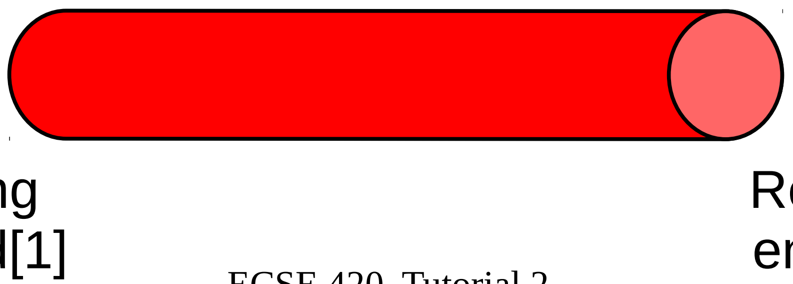
Part 3 – Another communication scheme

father

```
pid_t p, mypid;
int fd[2];
int n1, n2;
pipe(fd);
...
p = fork();
if (p == 0) {
  mypid = getpid();
  write(fd[1], &n1, sizeof(n1));
} else {
  read(fd[0], &n2, sizeof(n2));
}
```

child

```
pid_t p, mypid;
int fd[2];
int n1, n2;
pipe(fd);
...
p = fork();
if (p == 0) {
  mypid = getpid();
  write(fd[1], &n1, sizeof(n1));
} else {
  read(fd[0], &n2, sizeof(n2));
}
```



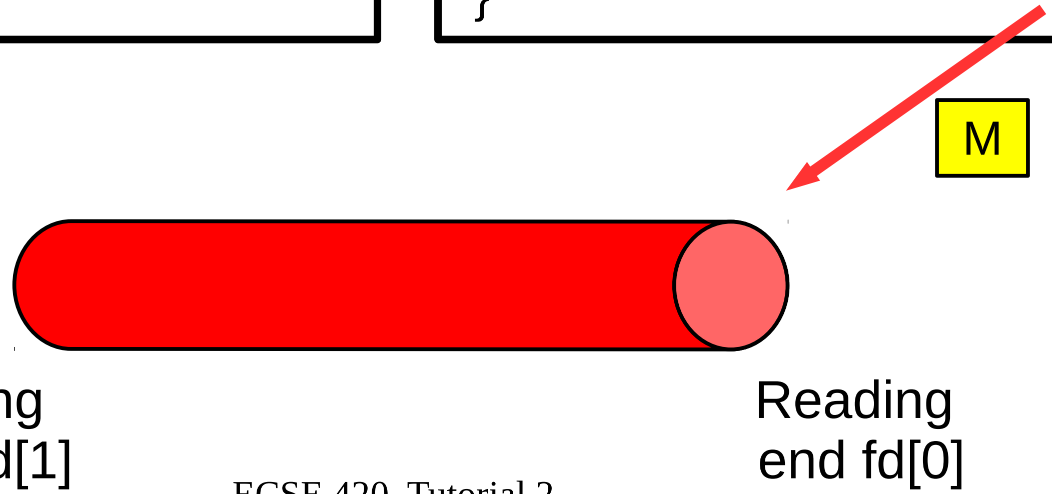
Part 3 – Another communication scheme

father

```
pid_t p, mypid;
int fd[2];
int n1, n2;
pipe(fd);
...
p = fork();
if (p == 0) {
  mypid = getpid();
  write(fd[1], &n1, sizeof(n1));
} else {
  read(fd[0], &n2, sizeof(n2));
}
```

child

```
pid_t p, mypid;
int fd[2];
int n1, n2;
pipe(fd);
...
p = fork();
if (p == 0) {
  mypid = getpid();
  write(fd[1], &n1, sizeof(n1));
} else {
  read(fd[0], &n2, sizeof(n2));
}
```



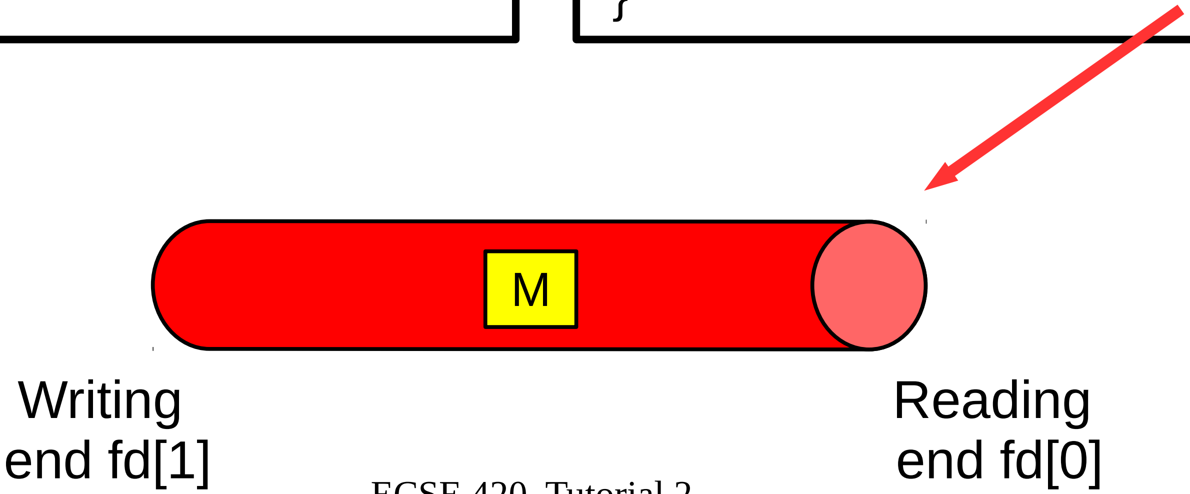
Part 3 – Another communication scheme

father

```
pid_t p, mypid;
int fd[2];
int n1, n2;
pipe(fd);
...
p = fork();
if (p == 0) {
  mypid = getpid();
  write(fd[1], &n1, sizeof(n1));
} else {
  read(fd[0], &n2, sizeof(n2));
}
```

child

```
pid_t p, mypid;
int fd[2];
int n1, n2;
pipe(fd);
...
p = fork();
if (p == 0) {
  mypid = getpid();
  write(fd[1], &n1, sizeof(n1));
} else {
  read(fd[0], &n2, sizeof(n2));
}
```



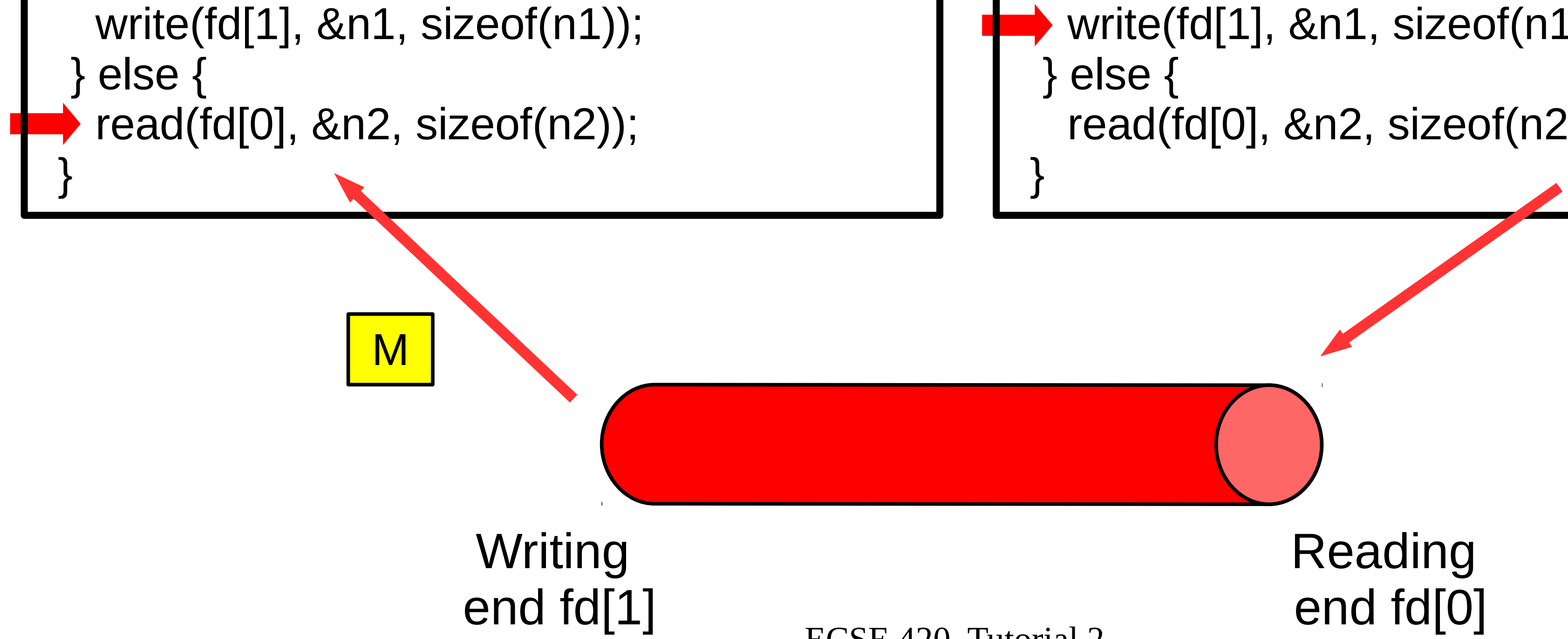
Part 3 – Another communication scheme

father

```
pid_t p, mypid;
int fd[2];
int n1, n2;
pipe(fd);
...
p = fork();
if (p == 0) {
  mypid = getpid();
  write(fd[1], &n1, sizeof(n1));
} else {
  read(fd[0], &n2, sizeof(n2));
}
```

child

```
pid_t p, mypid;
int fd[2];
int n1, n2;
pipe(fd);
...
p = fork();
if (p == 0) {
  mypid = getpid();
  write(fd[1], &n1, sizeof(n1));
} else {
  read(fd[0], &n2, sizeof(n2));
}
```



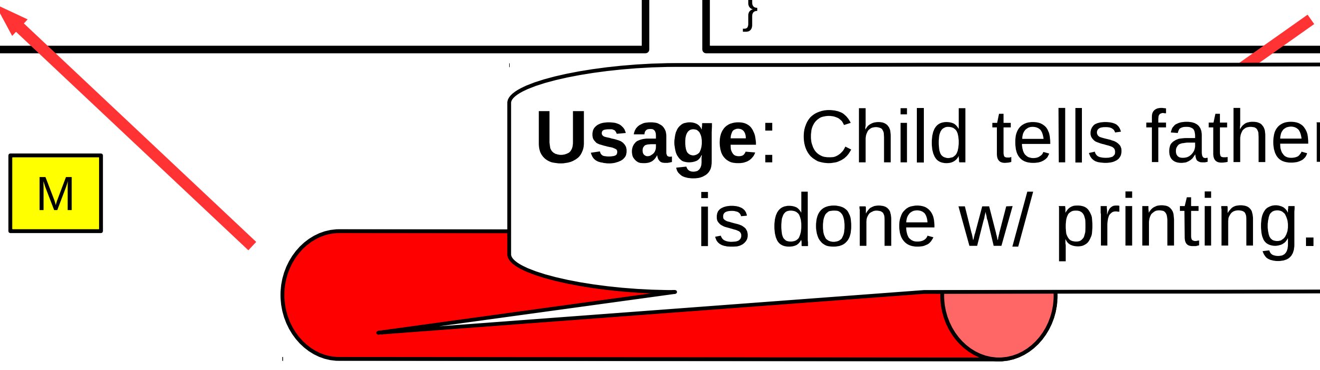
Part 3 – Another communication scheme

father

```
pid_t p, mypid;
int fd[2];
int n1, n2;
pipe(fd);
...
p = fork();
if (p == 0) {
  mypid = getpid();
  write(fd[1], &n1, sizeof(n1));
} else {
  read(fd[0], &n2, sizeof(n2));
}
```

child

```
pid_t p, mypid;
int fd[2];
int n1, n2;
pipe(fd);
...
p = fork();
if (p == 0) {
  mypid = getpid();
  write(fd[1], &n1, sizeof(n1));
} else {
  read(fd[0], &n2, sizeof(n2));
}
```



Usage: Child tells father that is done w/ printing.

Writing end fd[1]

Reading end fd[0]

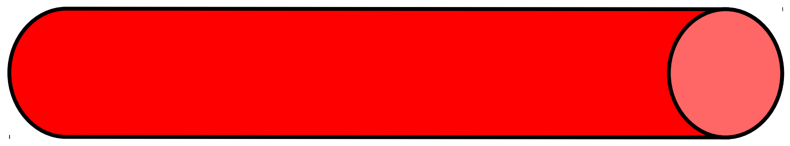
Part 3 – Empty pipe...

father

```
pid_t p, mypid;
int fd[2];
int n1, n2;
pipe(fd);
...
p = fork();
if (p == 0) {
  mypid = getpid();
  write(fd[1], &n1, sizeof(n1));
} else {
  read(fd[0], &n2, sizeof(n2));
}
```

child

```
pid_t p, mypid;
int fd[2];
int n1, n2;
pipe(fd);
...
p = fork();
if (p == 0) {
  mypid = getpid();
  write(fd[1], &n1, sizeof(n1));
} else {
  read(fd[0], &n2, sizeof(n2));
}
```



Part 3 – Empty pipe...

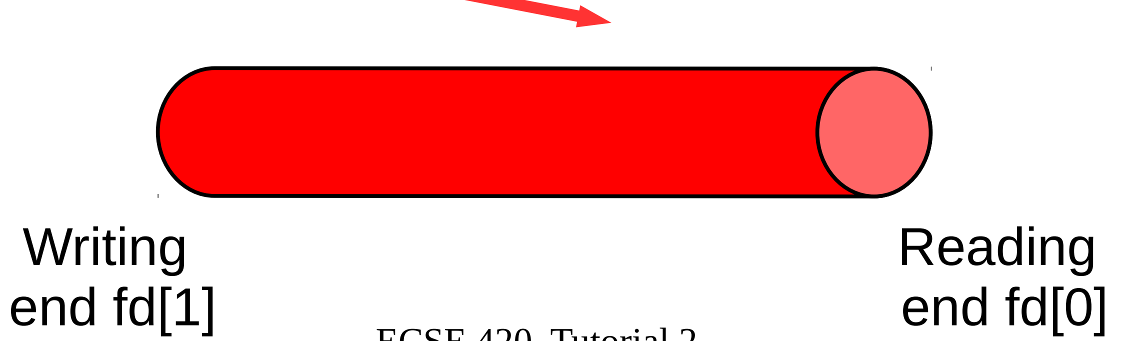
father

```
pid_t p, mypid;
int fd[2];
int n1, n2;
pipe(fd);
...
p = fork();
if (p == 0) {
  mypid = getpid();
  write(fd[1], &n1, sizeof(n1));
} else {
  read(fd[0], &n2, sizeof(n2));
}
```

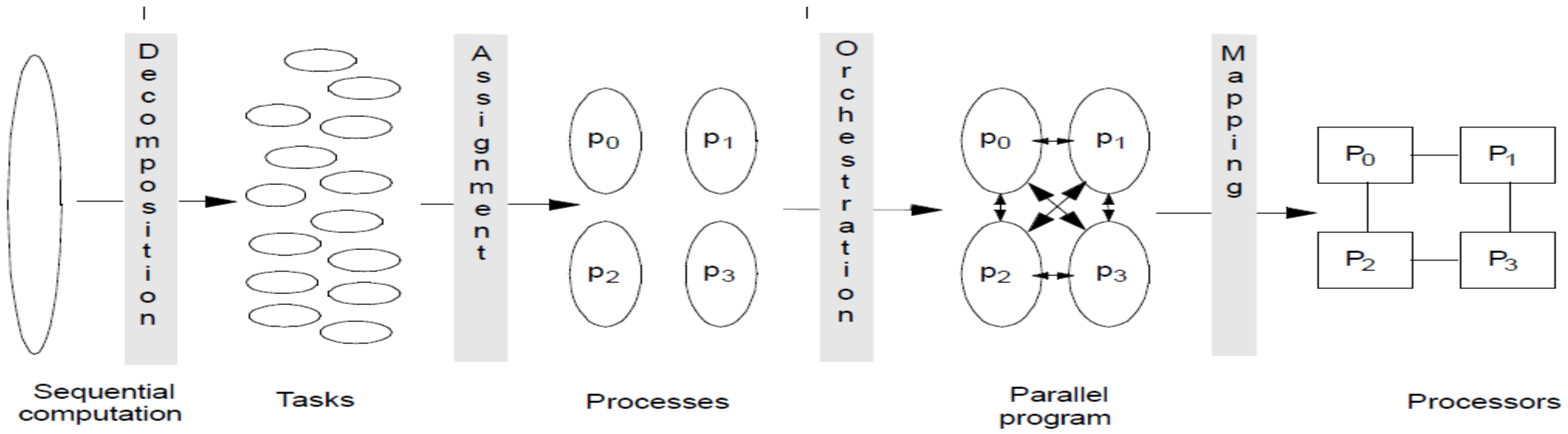
child

```
pid_t p, mypid;
int fd[2];
int n1, n2;
pipe(fd);
...
p = fork();
if (p == 0) {
  mypid = getpid();
  write(fd[1], &n1, sizeof(n1));
```

What will happen??

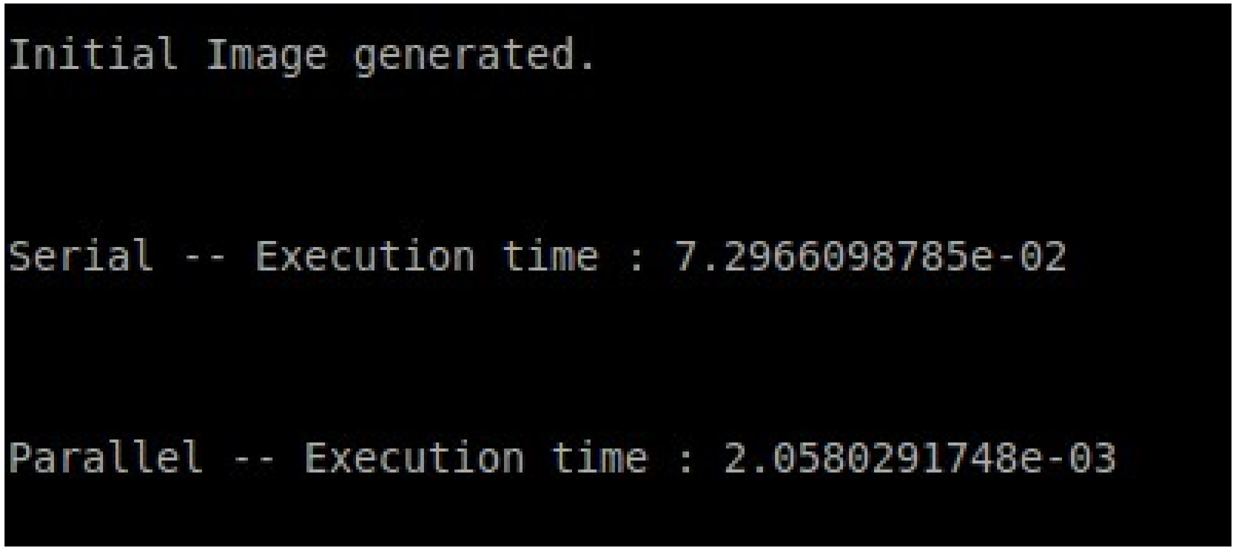


Introduction to Lab 1



1) Task to be performed – Image processing
2) Parallelization scheme – Processes
3) Synchronization scheme – Pipes
4) **Report results**

Part 4 – Results: Acceleration achieved

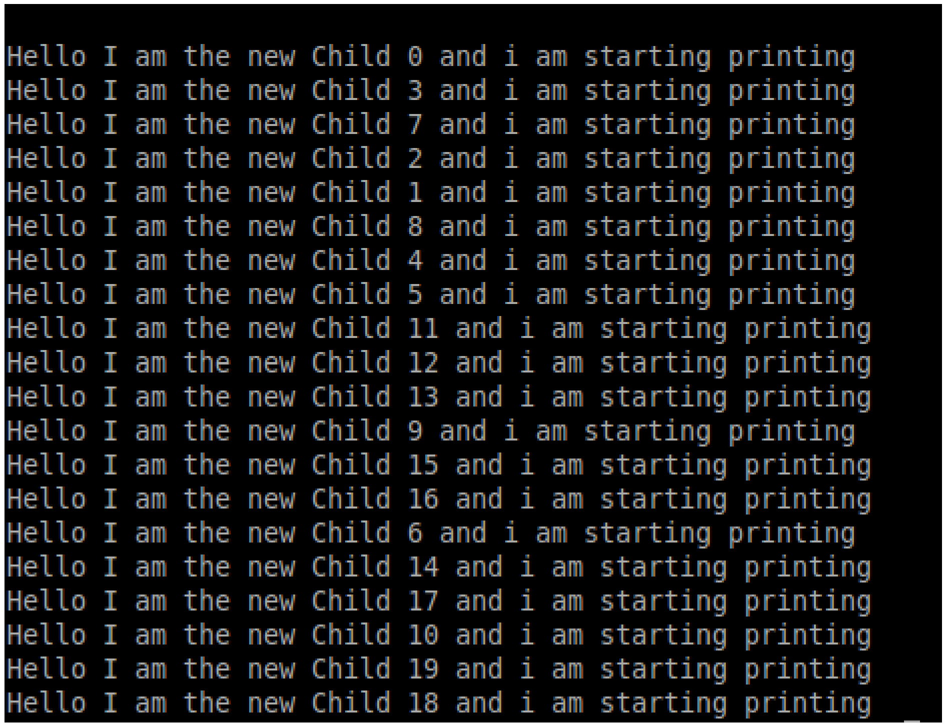


```
Initial Image generated.


Serial -- Execution time : 7.2966098785e-02


Parallel -- Execution time : 2.0580291748e-03
```

- Report results
- Explore speedup for different image sizes
- Explore how the number of processes affects speedup
- ...

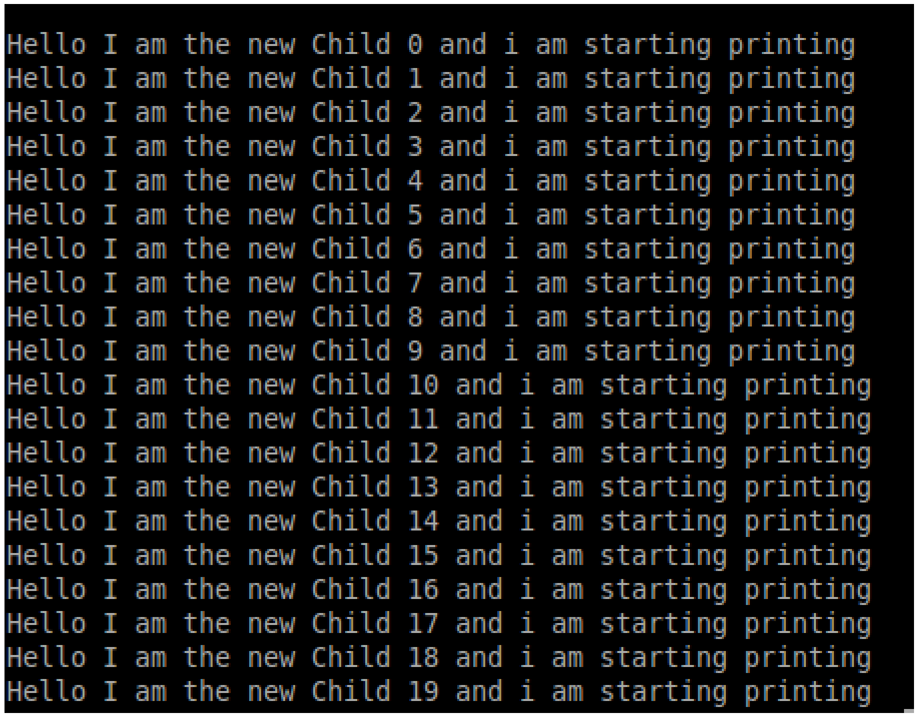
Part 4 – Results: Orchestration achieved

Before



```
Hello I am the new Child 0 and i am starting printing
Hello I am the new Child 3 and i am starting printing
Hello I am the new Child 7 and i am starting printing
Hello I am the new Child 2 and i am starting printing
Hello I am the new Child 1 and i am starting printing
Hello I am the new Child 8 and i am starting printing
Hello I am the new Child 4 and i am starting printing
Hello I am the new Child 5 and i am starting printing
Hello I am the new Child 11 and i am starting printing
Hello I am the new Child 12 and i am starting printing
Hello I am the new Child 13 and i am starting printing
Hello I am the new Child 9 and i am starting printing
Hello I am the new Child 15 and i am starting printing
Hello I am the new Child 16 and i am starting printing
Hello I am the new Child 6 and i am starting printing
Hello I am the new Child 14 and i am starting printing
Hello I am the new Child 17 and i am starting printing
Hello I am the new Child 10 and i am starting printing
Hello I am the new Child 19 and i am starting printing
Hello I am the new Child 18 and i am starting printing
```

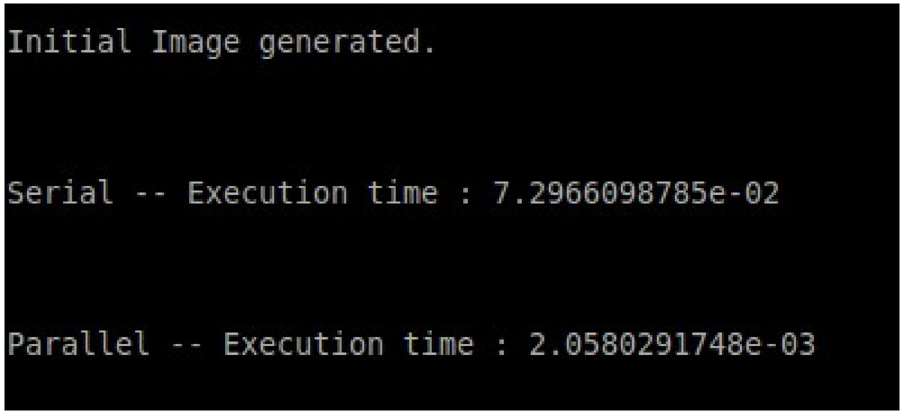
After



```
Hello I am the new Child 0 and i am starting printing
Hello I am the new Child 1 and i am starting printing
Hello I am the new Child 2 and i am starting printing
Hello I am the new Child 3 and i am starting printing
Hello I am the new Child 4 and i am starting printing
Hello I am the new Child 5 and i am starting printing
Hello I am the new Child 6 and i am starting printing
Hello I am the new Child 7 and i am starting printing
Hello I am the new Child 8 and i am starting printing
Hello I am the new Child 9 and i am starting printing
Hello I am the new Child 10 and i am starting printing
Hello I am the new Child 11 and i am starting printing
Hello I am the new Child 12 and i am starting printing
Hello I am the new Child 13 and i am starting printing
Hello I am the new Child 14 and i am starting printing
Hello I am the new Child 15 and i am starting printing
Hello I am the new Child 16 and i am starting printing
Hello I am the new Child 17 and i am starting printing
Hello I am the new Child 18 and i am starting printing
Hello I am the new Child 19 and i am starting printing
```

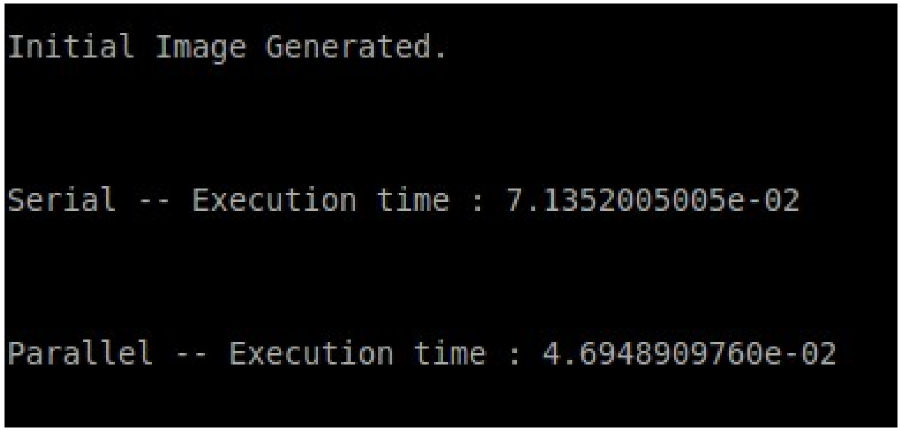
Part 4 – Results: Communication overhead

Without pipes



```
Initial Image generated.


Serial -- Execution time : 7.2966098785e-02


Parallel -- Execution time : 2.0580291748e-03
```

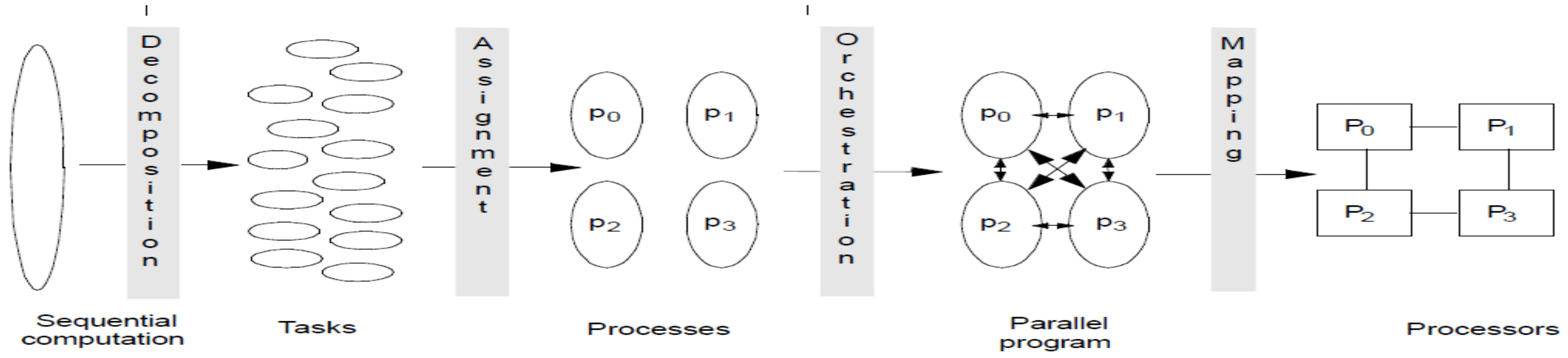
With pipes



```
Initial Image Generated.


Serial -- Execution time : 7.1352005005e-02


Parallel -- Execution time : 4.6948909760e-02
```

Lab 1

- Task to be performed – Image processing
- Parallelization scheme – Processes
- Synchronization scheme – Pipes
- Report results



- Will be posted next week.
- Code examples will be attached as well.
- More detailed updates during the following days.

Lab 1 – Any questions??