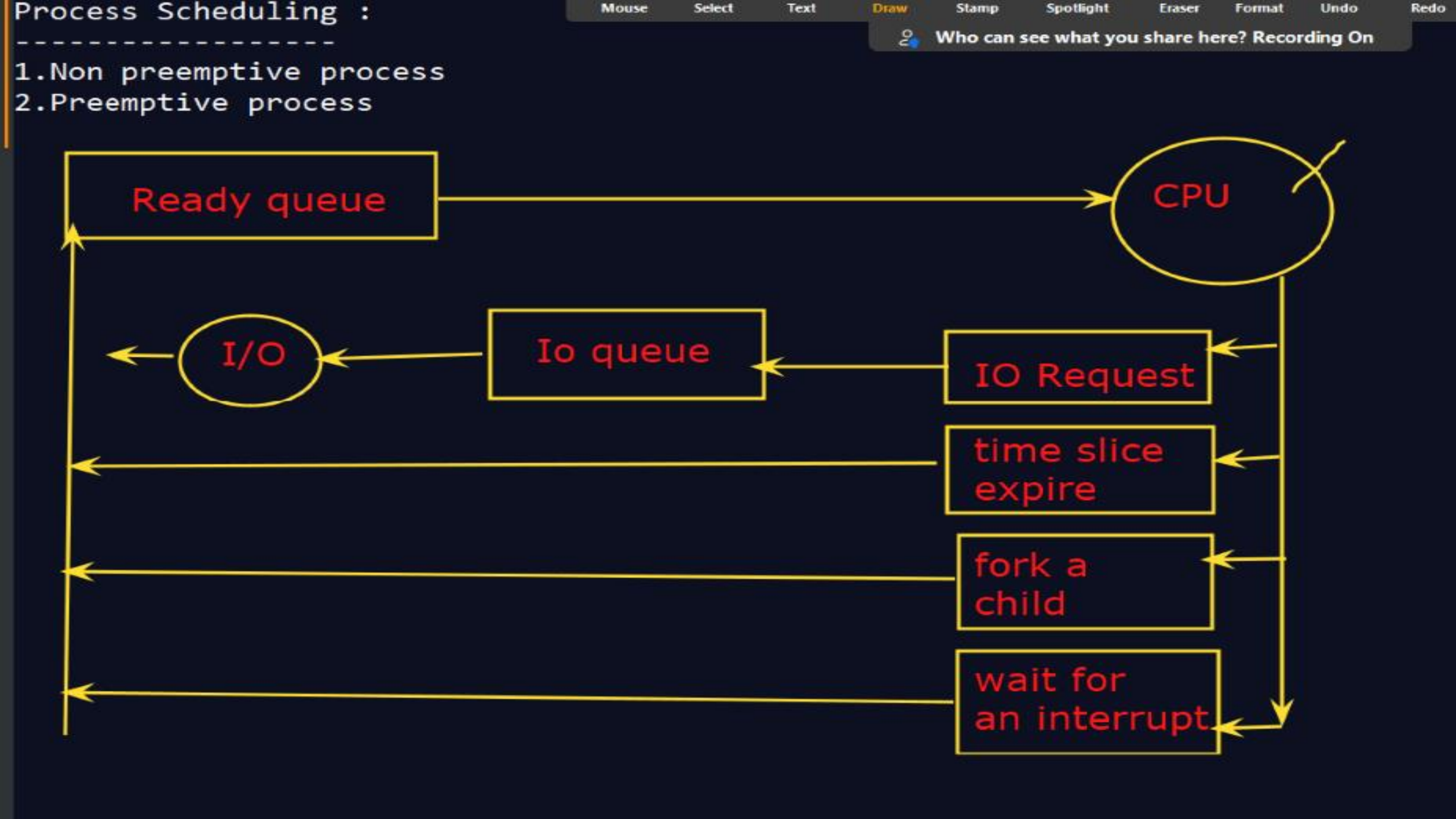
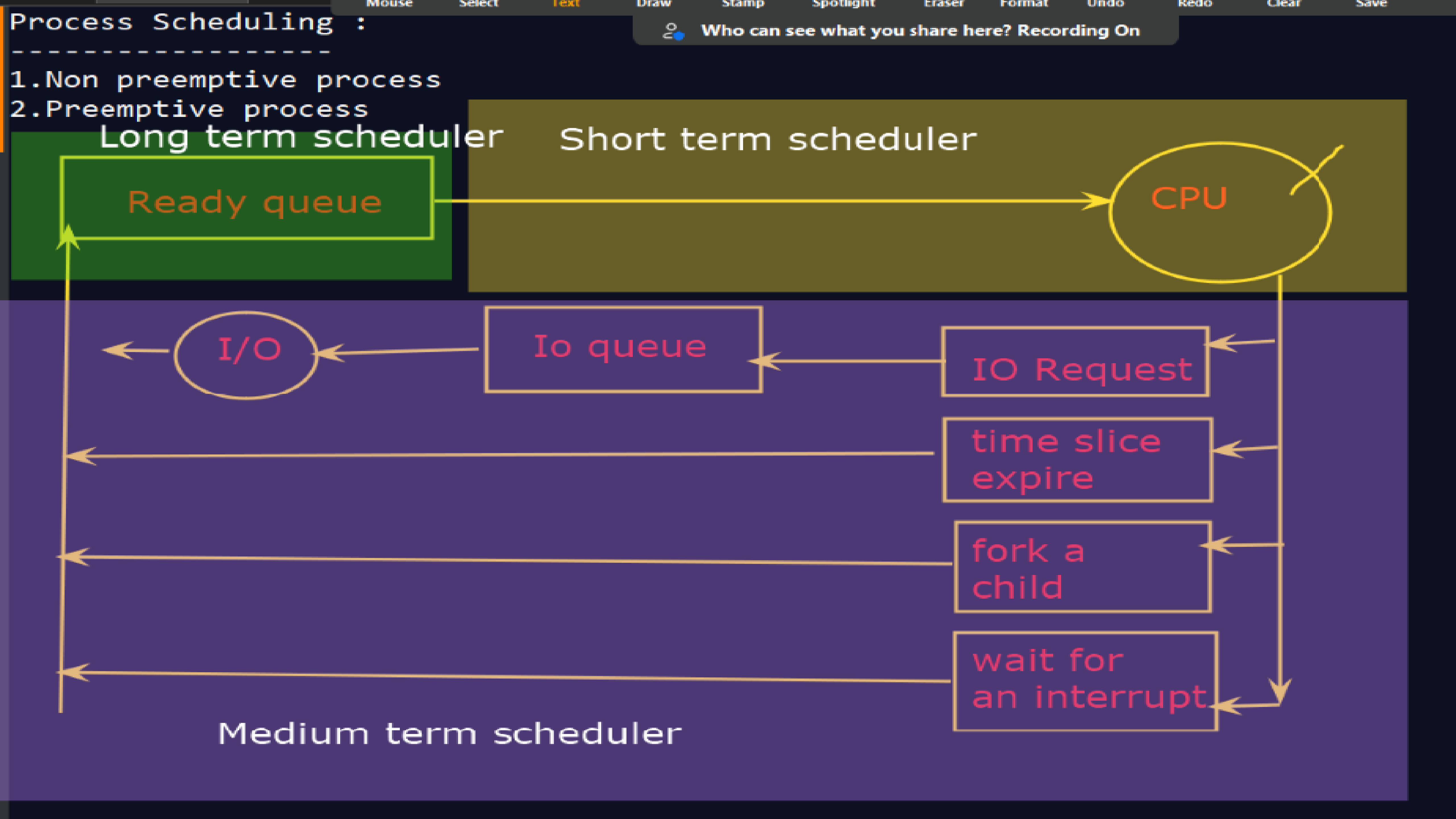


Process Scheduling :

1.Non preemptive process

2.Preemptive process



Process Scheduling :

1.Non preemptive process
2.Preemptive process

Long term scheduler

Ready queue

Short term scheduler

CPU

I/O

Io queue

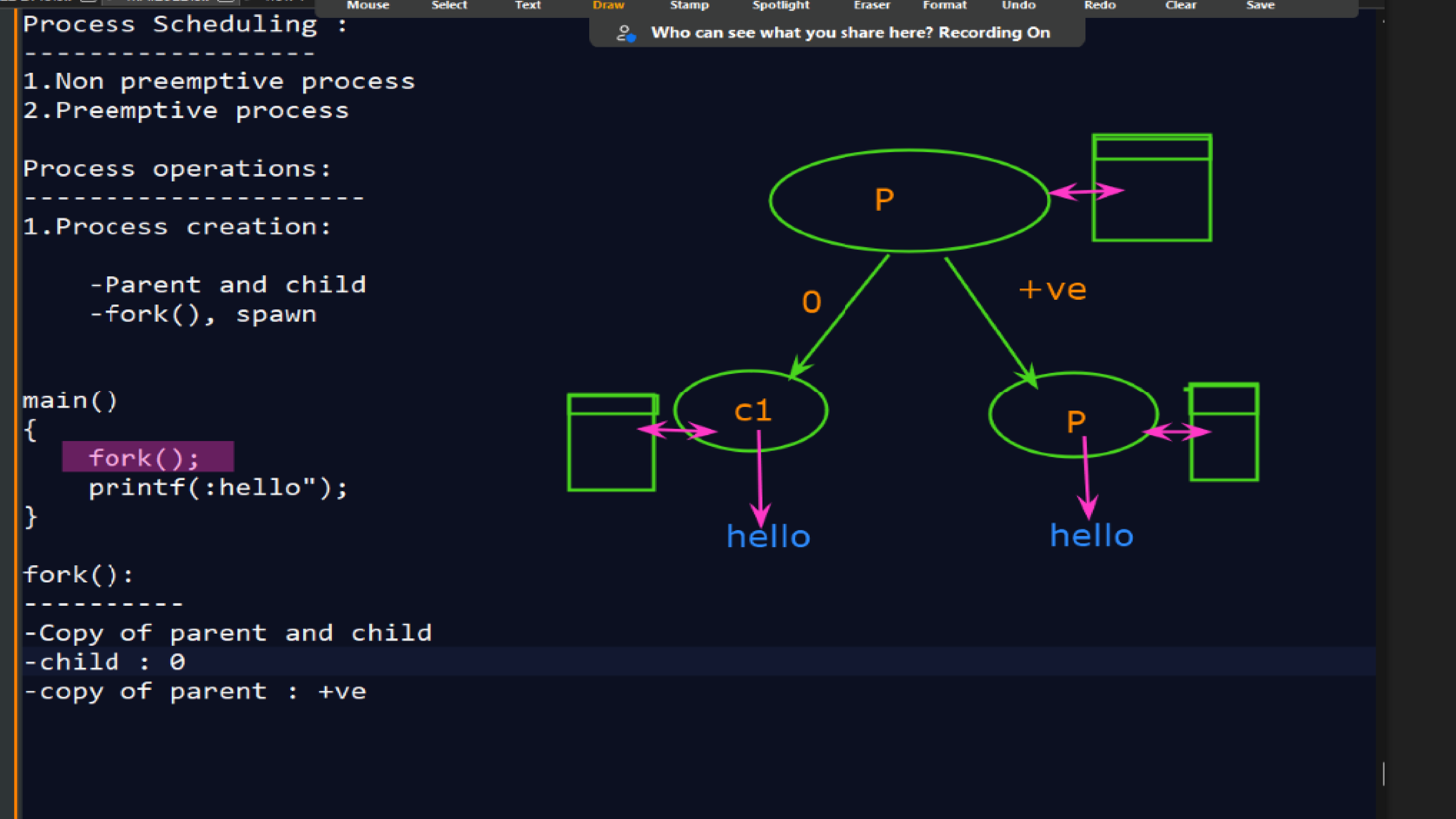
IO Request

time slice expire

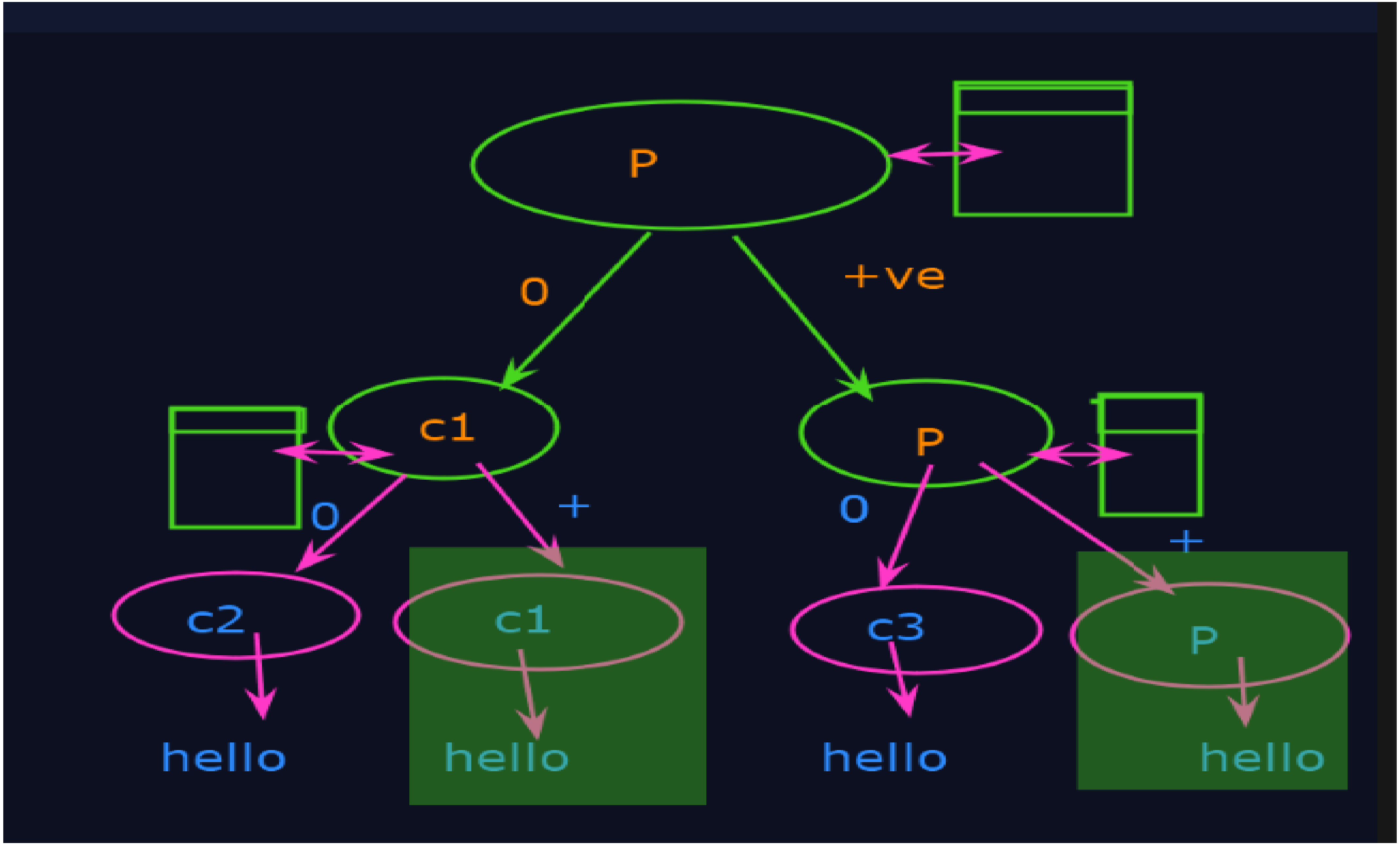
fork a child

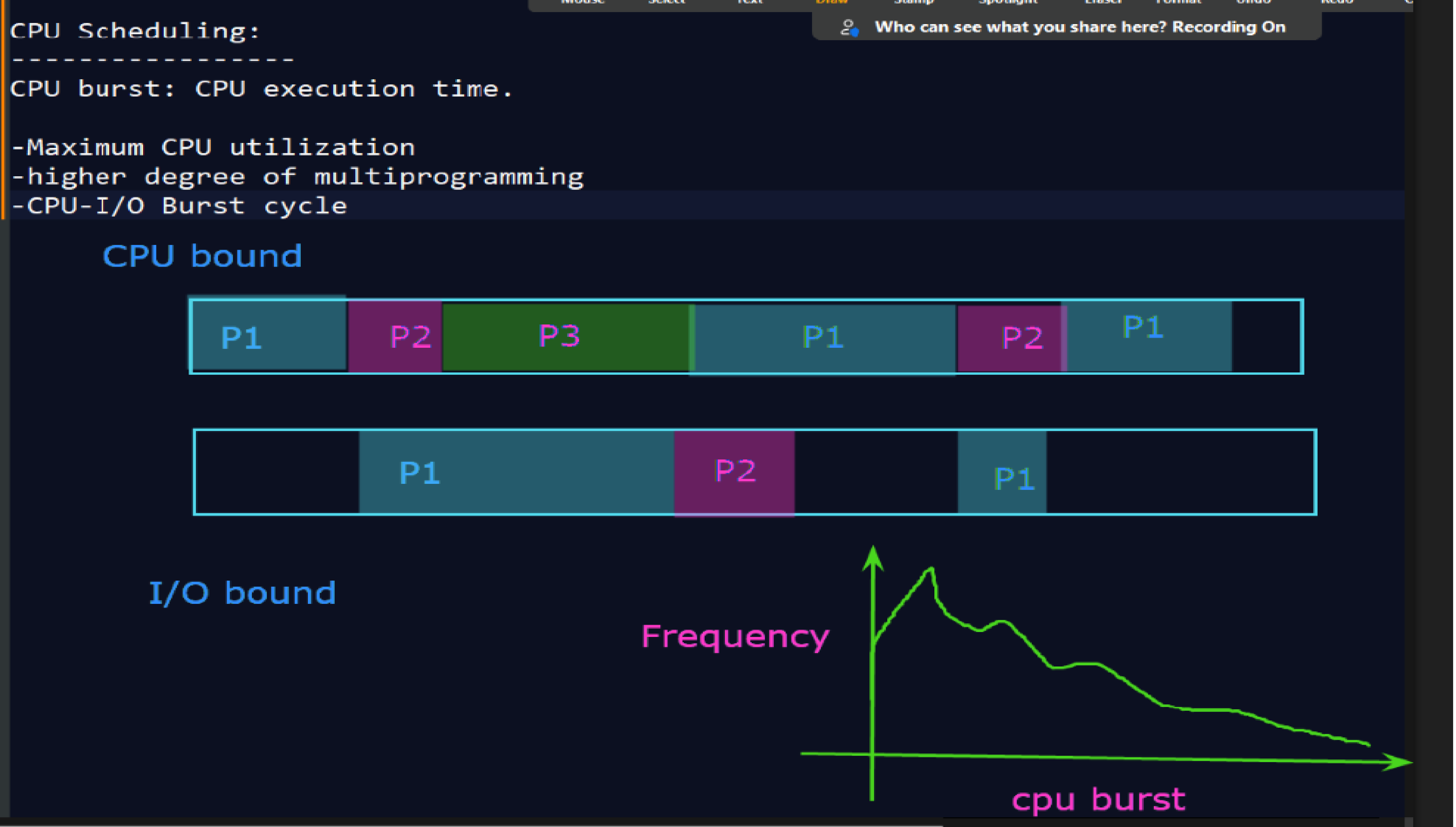
wait for an interrupt

Medium term scheduler



```
Process Scheduling :
------------------
1.Non preemptive process
2.Preemptive process

Process operations:
---------------------
1.Process creation:

    -Parent and child
    -fork(), spawn


main()
{
    fork();
    printf(:hello");
}

fork():
----------
-Copy of parent and child
-child : 0
-copy of parent : +ve
```


P
0
+ve
c1
P
0
+
0
+
c2
c1
c3
P
hello
hello
hello
hello



CPU Scheduling:

CPU burst: CPU execution time.

-Maximum CPU utilization
-higher degree of multiprogramming
-CPU-I/O Burst cycle

Chapter 5: CPU Scheduling

- Basic Concepts
- Scheduling Criteria
- Scheduling Algorithms
- Thread Scheduling
- Multiple-Processor Scheduling
- Operating Systems Examples
- Algorithm Evaluation

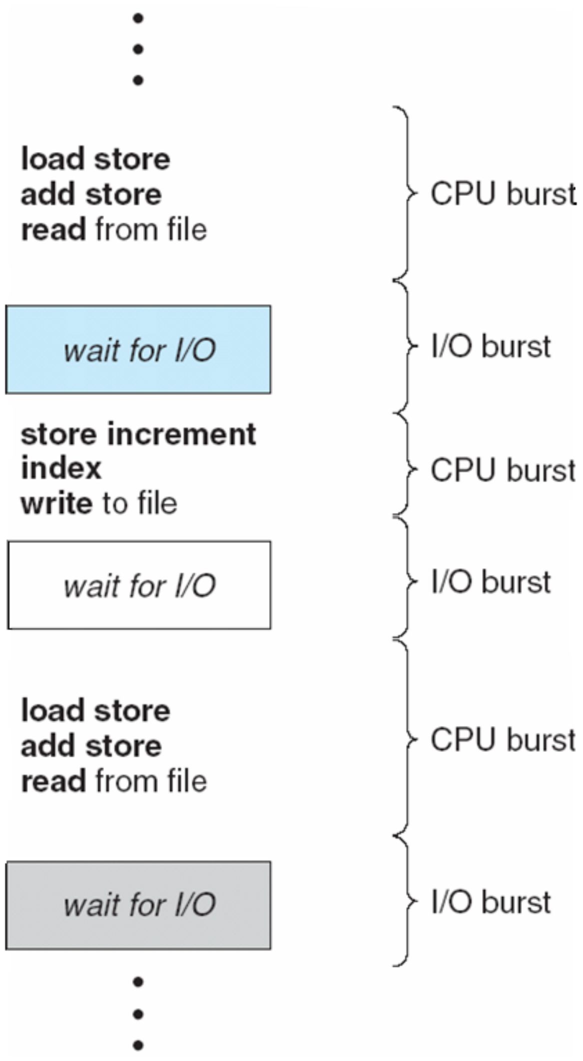
Objectives

- To introduce CPU scheduling, which is the basis for multiprogrammed operating systems
- To describe various CPU-scheduling algorithms
- To discuss evaluation criteria for selecting a CPU-scheduling algorithm for a particular system

Basic Concepts

- Maximum CPU utilization obtained with multiprogramming
- CPU–I/O Burst Cycle – Process execution consists of a *cycle* of CPU execution and I/O wait
- **CPU burst** distribution

Alternating Sequence of CPU And I/O Bursts



CPU Scheduler

- Selects from among the **processes in memory that are ready to execute**, and allocates the CPU to one of them
- CPU scheduling decisions may take place when a process:
  1. Switches from **running to waiting state**
  2. Switches from **running to ready state**
  3. Switches from **waiting to ready**
  4. Terminates
- Scheduling under 1 and 4 is **nonpreemptive**
- All other scheduling is **preemptive**

Dispatcher

- Dispatcher module gives control of the CPU to the process selected by the short-term scheduler; this involves:
  - switching context
  - switching to user mode
  - jumping to the proper location in the user program to restart that program
- **Dispatch latency** – time it takes for the dispatcher to stop one process and start another running

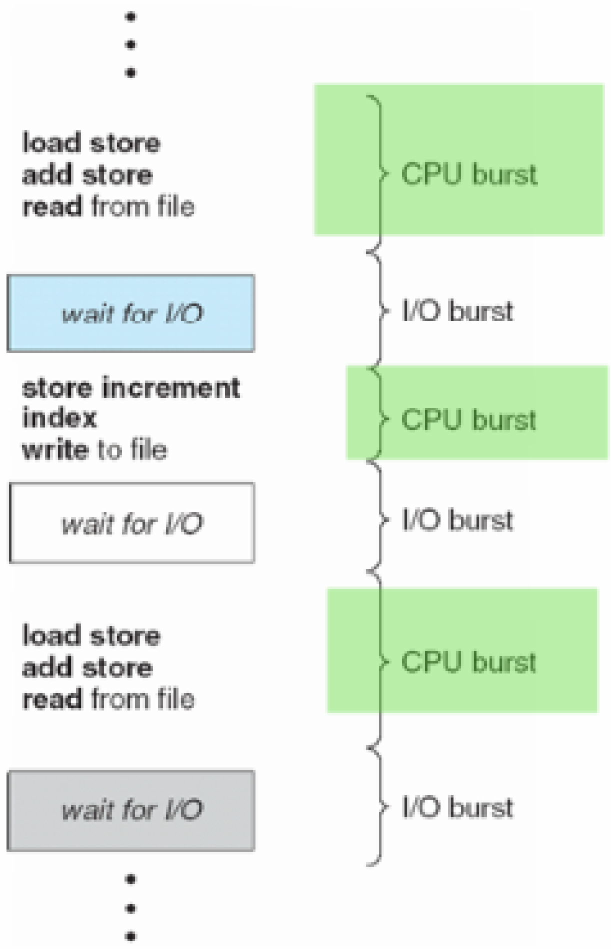
Scheduling Criteria

- **CPU utilization** – keep the CPU as busy as possible
- **Throughput** – # of processes that complete their execution per time unit
- **Turnaround time** – amount of time to execute a particular process
- **Waiting time** – amount of time a process has been waiting in the ready queue
- **Response time** – amount of time it takes from when a request was submitted until the first response is produced, not output (for time-sharing environment)

Scheduling Algorithm Optimization Criteria

- Max CPU utilization
- Max throughput
- Min turnaround time
- Min waiting time
- Min response time

Alternating Sequence of CPU And I/O Bursts



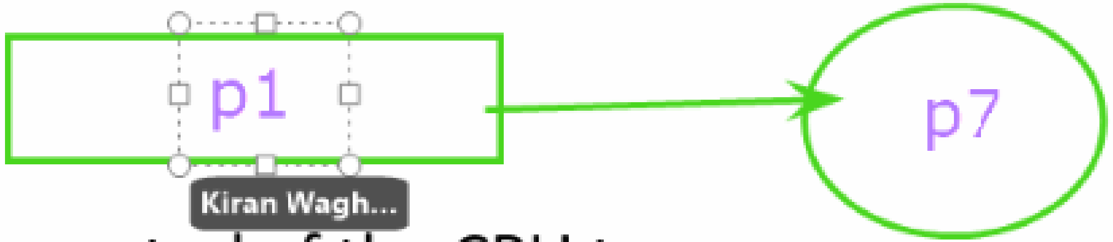
1. Running -> waiting
2.Running -> Ready
3.Waiting ->Ready

Scheduler

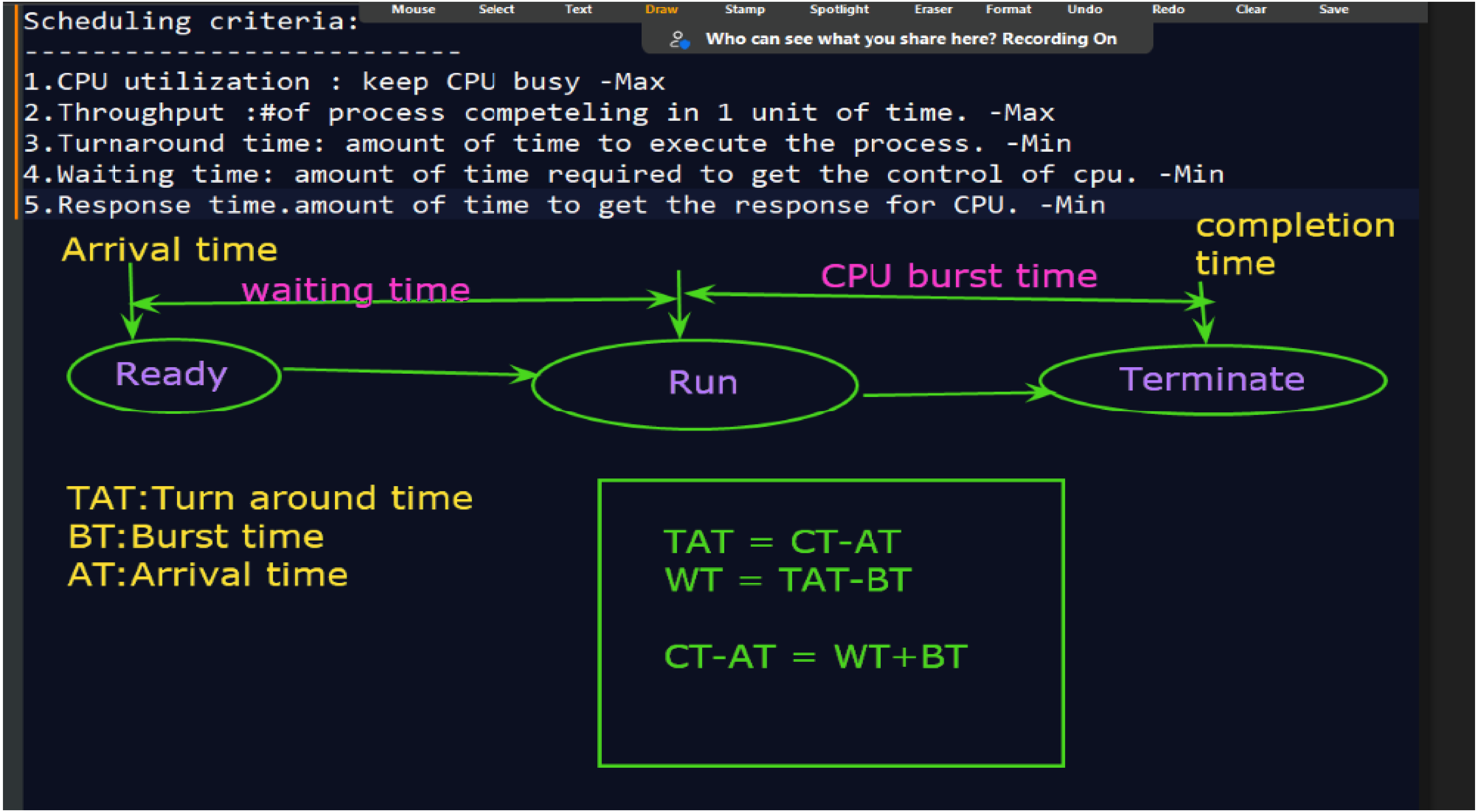
↓

Scheduling algorithms

Dispatcher



- Dispatcher module gives control of the CPU to the process selected by the short-term scheduler; this involves:
    - switching context
    - switching to user mode
    - jumping to the proper location in the user program to restart that program
- **Dispatch latency** – time it takes for the dispatcher to stop one process and start another running

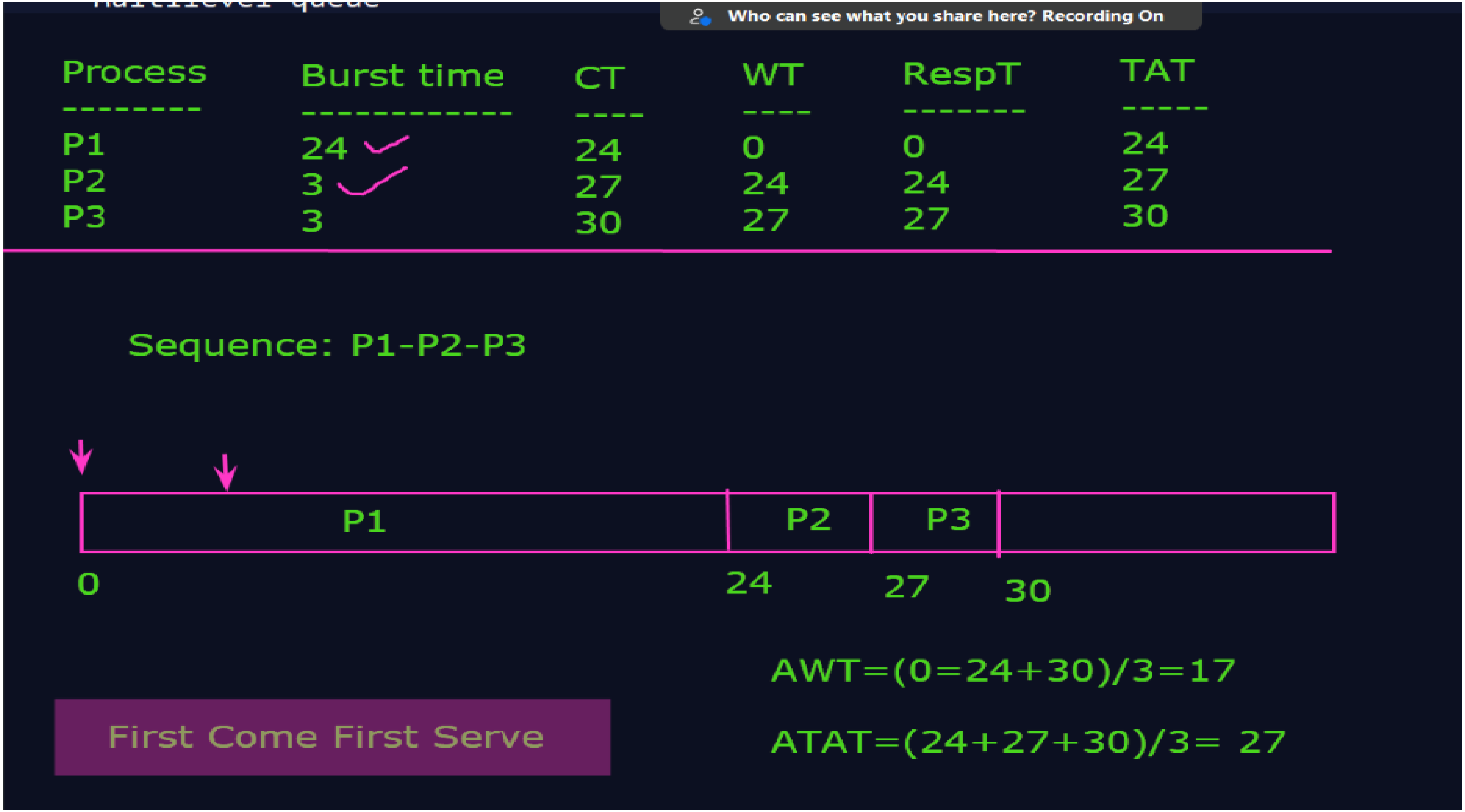


```
Scheduling criteria:
---------------------------
1.CPU utilization : keep CPU busy -Max
2.Throughput :#of process competeling in 1 unit of time. -Max
3.Turnaround time: amount of time to execute the process. -Min
4.Waiting time: amount of time required to get the control of cpu. -Min
5.Response time.amount of time to get the response for CPU. -Min
```

TAT:Turn around time
BT:Burst time
AT:Arrival time

TAT = CT-AT
WT = TAT-BT

CT-AT = WT+BT



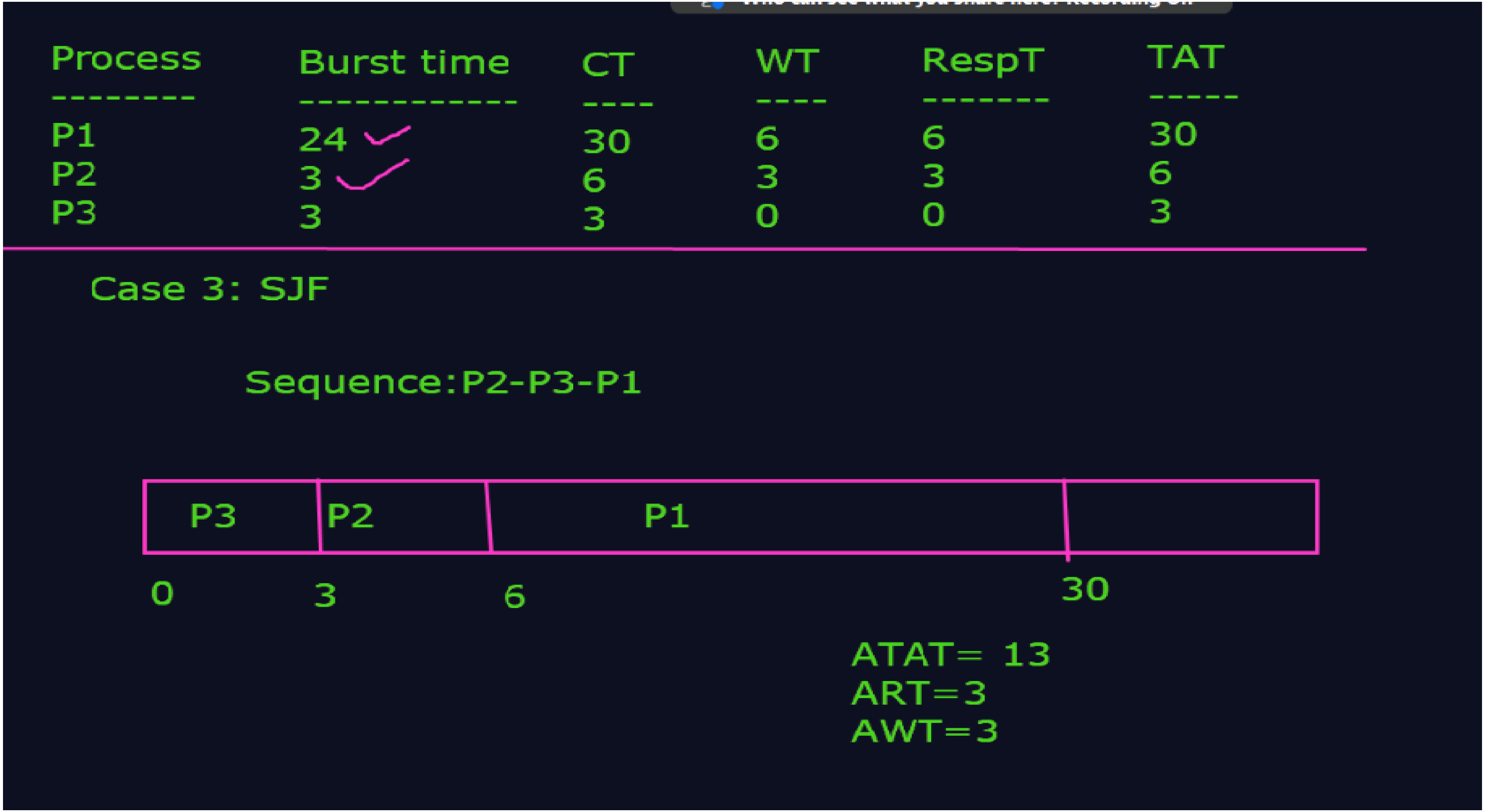
Process	Burst time	CT	WT	RespT	TAT
P1	24 ✓	24	0	0	24
P2	3 ✓	27	24	24	27
P3	3	30	27	27	30

Sequence: P1-P2-P3

AWT=(0=24+30)/3=17

ATAT=(24+27+30)/3= 27

First Come First Serve



Process	Burst time	CT	WT	RespT	TAT
P1	24 ✓	30	6	6	30
P2	3 ✓	6	3	3	6
P3	3	3	0	0	3

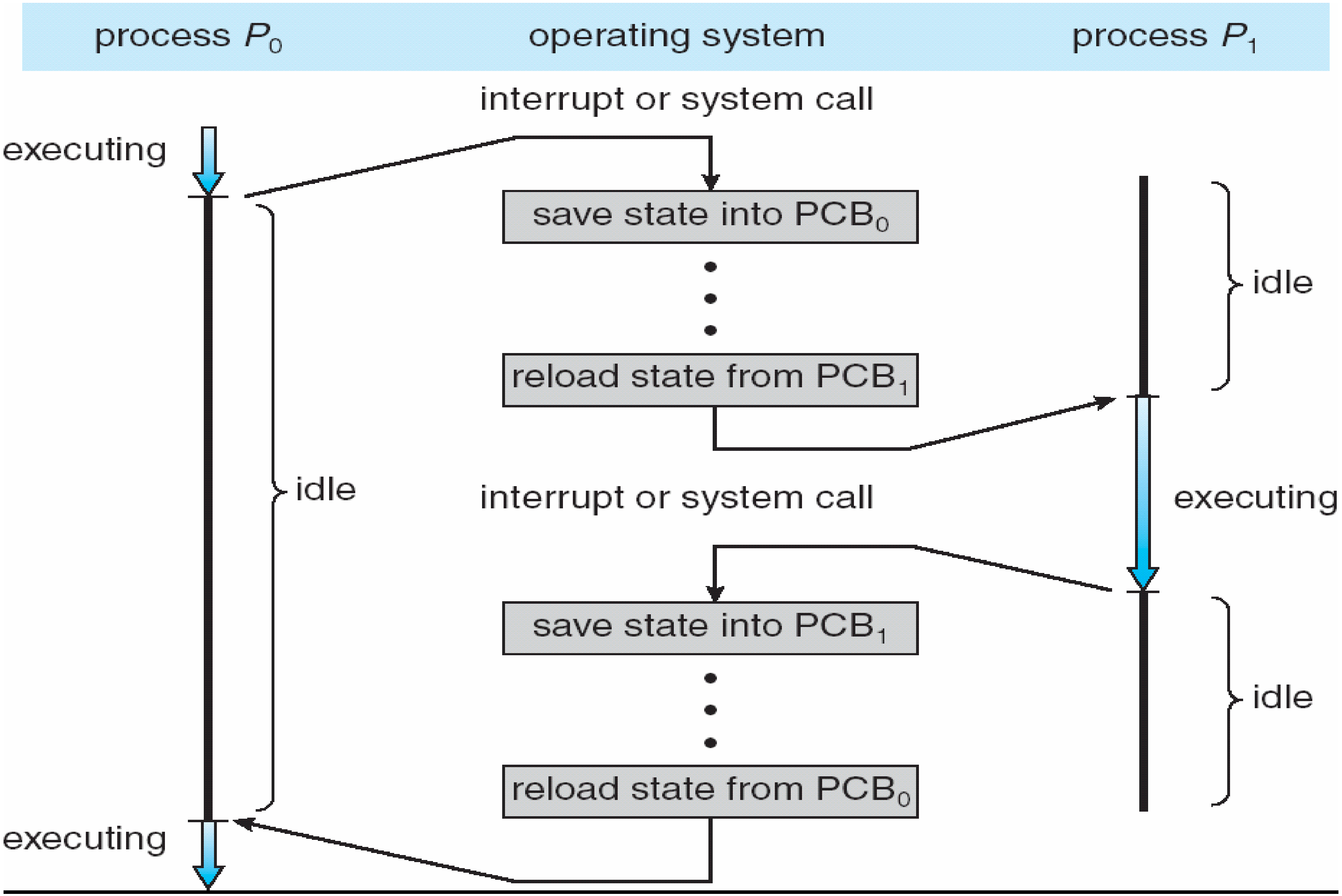
Case 3: SJF

Sequence:P2-P3-P1

P3	P2	P1	

0 3 6 30

ATAT= 13
ART=3
AWT=3


process P_0
operating system
process P_1
interrupt or system call
executing
save state into PCB_0
idle
reload state from PCB_1
idle
interrupt or system call
executing
save state into PCB_1
idle
reload state from PCB_0
executing