Process Scheduling Queue

Note Title 22/02/2007

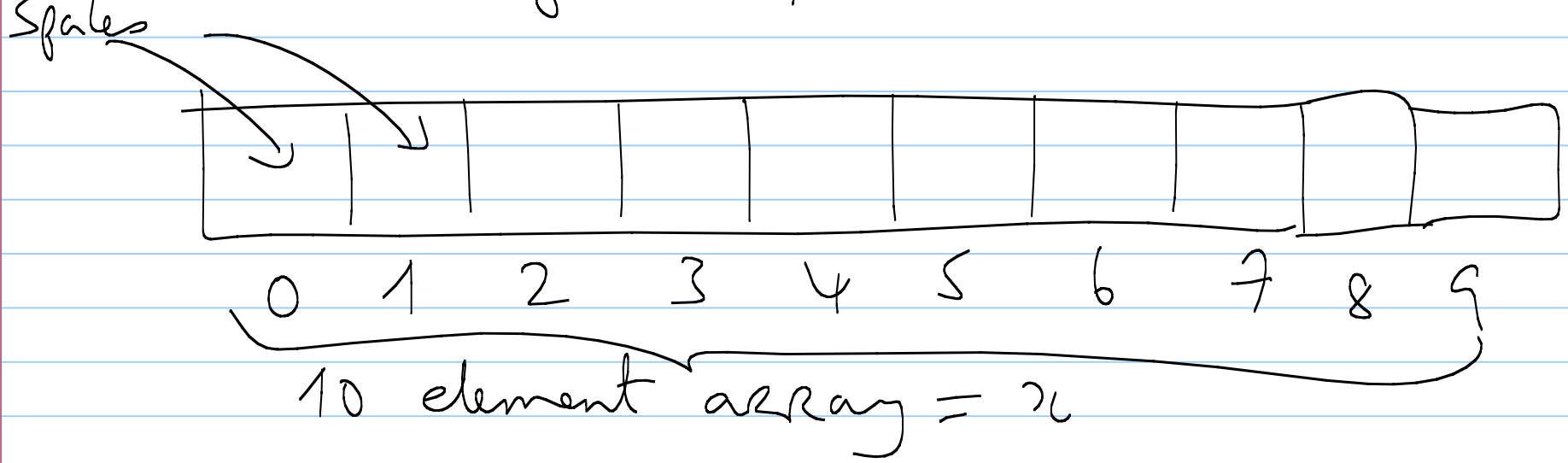
How do I program a queue?

Possible Key elements of a queue:

- Index
- Priority
- Parameters (size, elements, etc.)
- Scheduler ??
- Head of a queue
- Tail of a queue
- member or element in a queue

Possible Solns:

(A) **Array:** set of spaces / boxes, each having a designated position-number



How to label it?

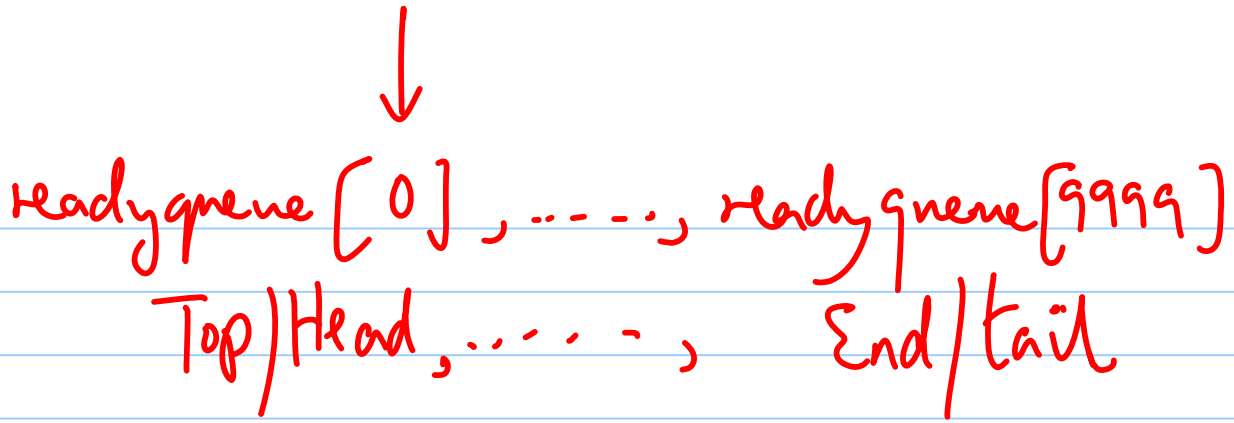
for array x : $x_0, x_1, x_2, \ldots x_9$

In programming terms : $x[0], x[1], x[2], \ldots, x[9]$

Foreg: Declare an array of 10 integers:

int x[10]; ← x[0], x[1], ..., x[9]

process readyqueue[10000];



Problems with using Arrays to program a queue:

- May not allocate enough elements (not enough memory allocated)
- May allocate too many elements (memory space wasted because it won't be used.)

(B) Linked List : (or any data structure)

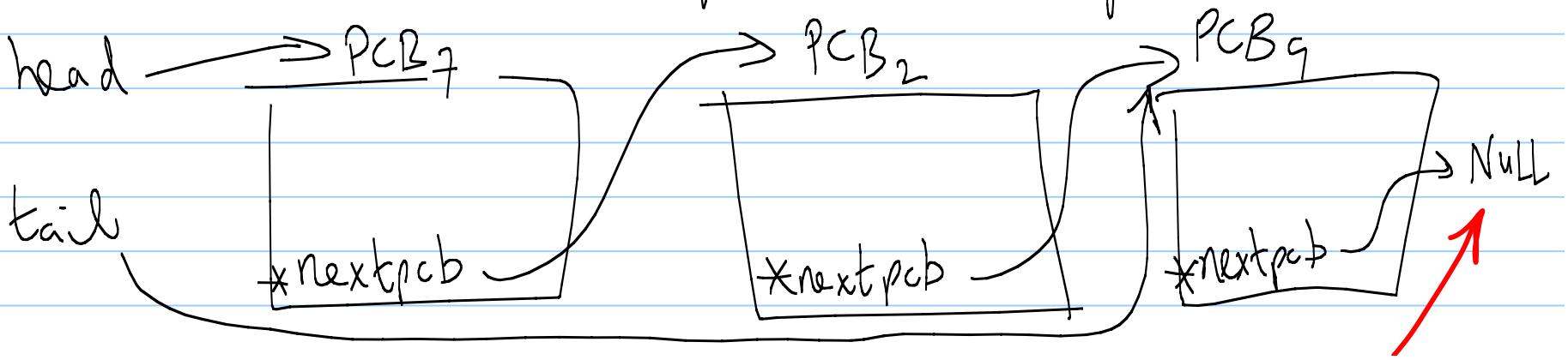
Mechanism for connecting / linking processes WITHOUT using an index or predefined size for the queue

∴ We need to define:

- head of the queue
- tail of the queue
- a method of connecting one process to the next one in the queue, and so on.

This creates a linked list or chain

The connection is achieved by each PCB (which represents a process) having a variable called nextpcb, which contains the address of the next PCB in the queue. Example



Some facts about this example:

tail → nextpcb = NULL;

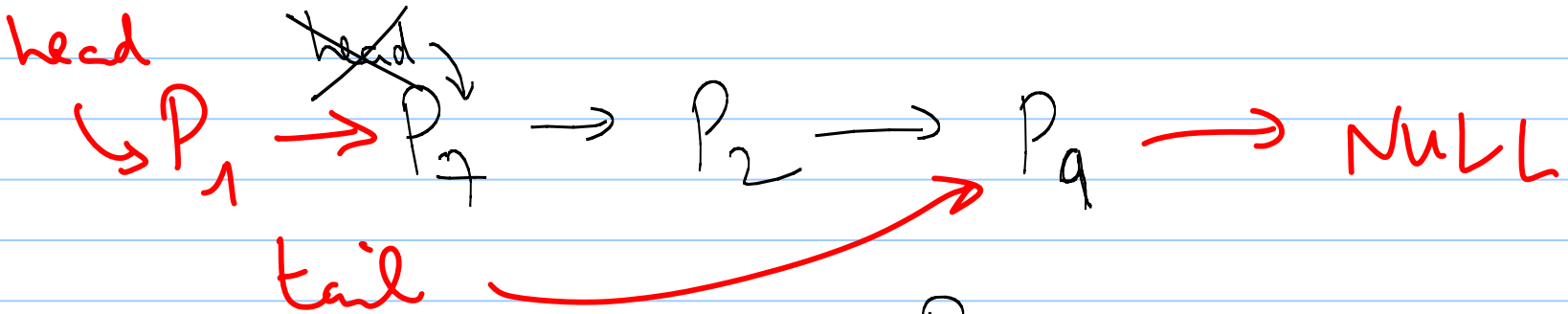
Say the OS wants to add in a new Process, P_1:

There are three possible places to "insert" into the queue:

1. head : if it's the highest priority process
2. tail : if it's the lowest priority process
3. Somewhere between the head and the tail depending

on its priority.

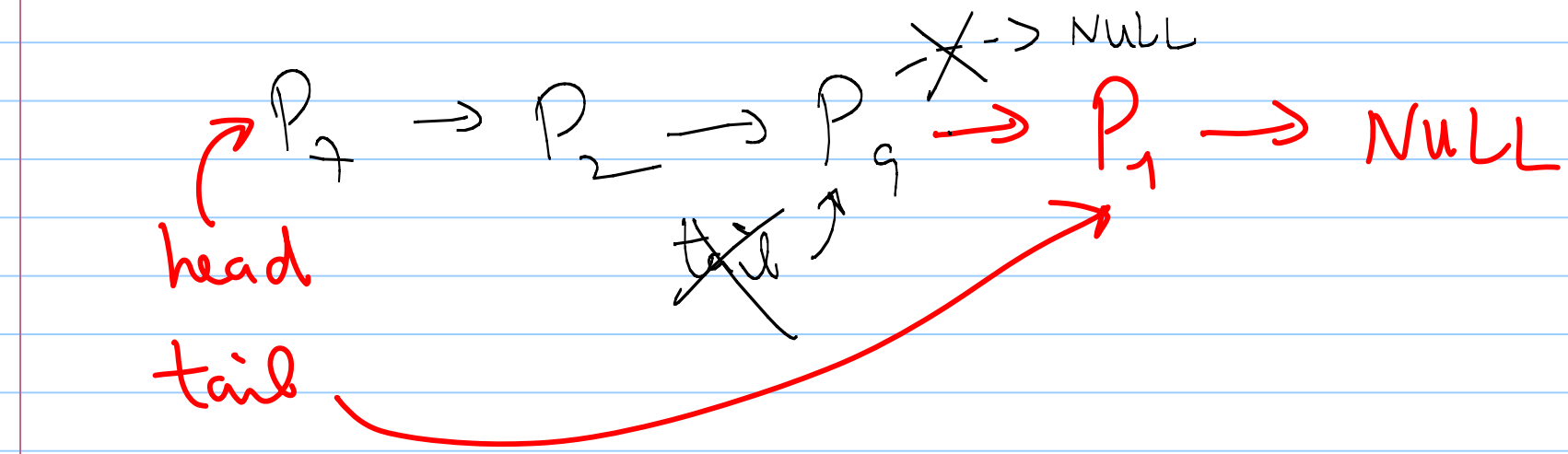
Case ① : P_1 becomes the new head.



newp = our new process, P_1:

```
newp -> nextpcb = head;   // old head
head = newp;              // setting the new head
```

Case ② : P_1 becomes the new tail :



$P_9 \rightarrow$ ~~nextptr~~ $= P_1$

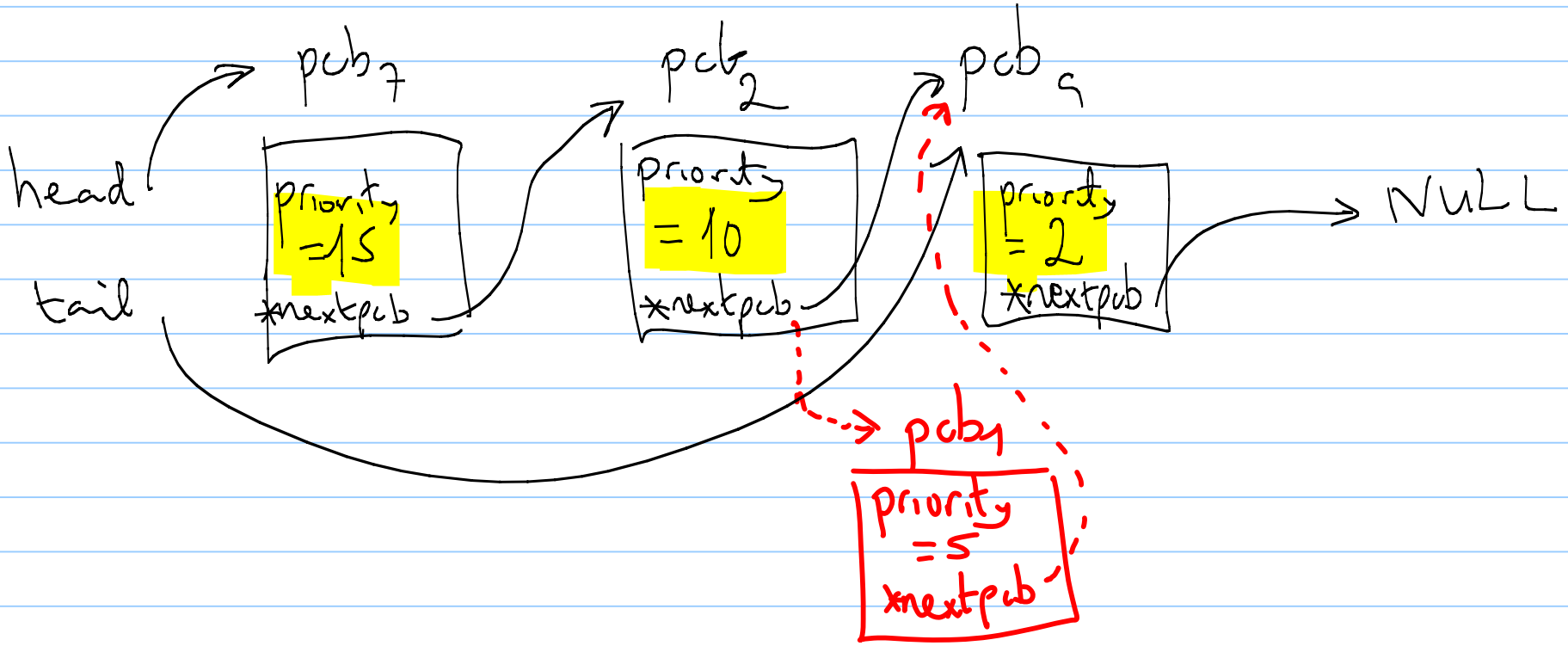
We have no index !!!

```
// newp is new process, P1.

for( p=head; p->nextpcb=NULL; p = p->nextpcb )
{
    // test to See if p -> nextpcb = NULL
    // if so p->nextpcb = newp;

    if( p->nextpcb == NULL)
    {
        tail = newp;
        newp-> nextpcb = NULL;
    }
}
```

Case 3: P_1 goes somewhere between head & tail:



Use a loop:

```
for (p=head; p->nextpcb=NULL; p=p->nextpcb)
{
    //Check priority value of newp
    if((newp.priority<p.priority)&&(newp.priority>(p->nextpcb).priority))
    {
    PCB tempp; //declare tempP to hold "old" value of p->nextpcb
    tempp = p->nextpcb; //Store "old" p's nextpcb
    p->nextpcb = newp;
    newp->nextpcb = tempp;
    }
}
```

Putting it all together

Reminder: Case ① : newp becomes new head

Case ② : newp becomes new tail

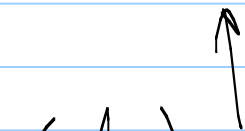
case ③ : newp inserted somewhere in between

Code follows on next page

```
// Finally!
if (newp.priority > head.priority)
{
    //Case 1 code as above
}
else if (newp.priority < tail.priority)
{
    // Case 2 code as above
}
else{
    // case 3 code as above
}
```

This is a programming method for implementing a scheduling queue.

It uses a singly linked list (simplest type of linked list)



(links are in one direction)