CSCI-3753: Operating Systems
Spring 2021

Biljith Thadichi

Department of Computer Science

University of Colorado Boulder

Based on slides by Abigail Fernandes

Week 12
> Programming Assignment 4

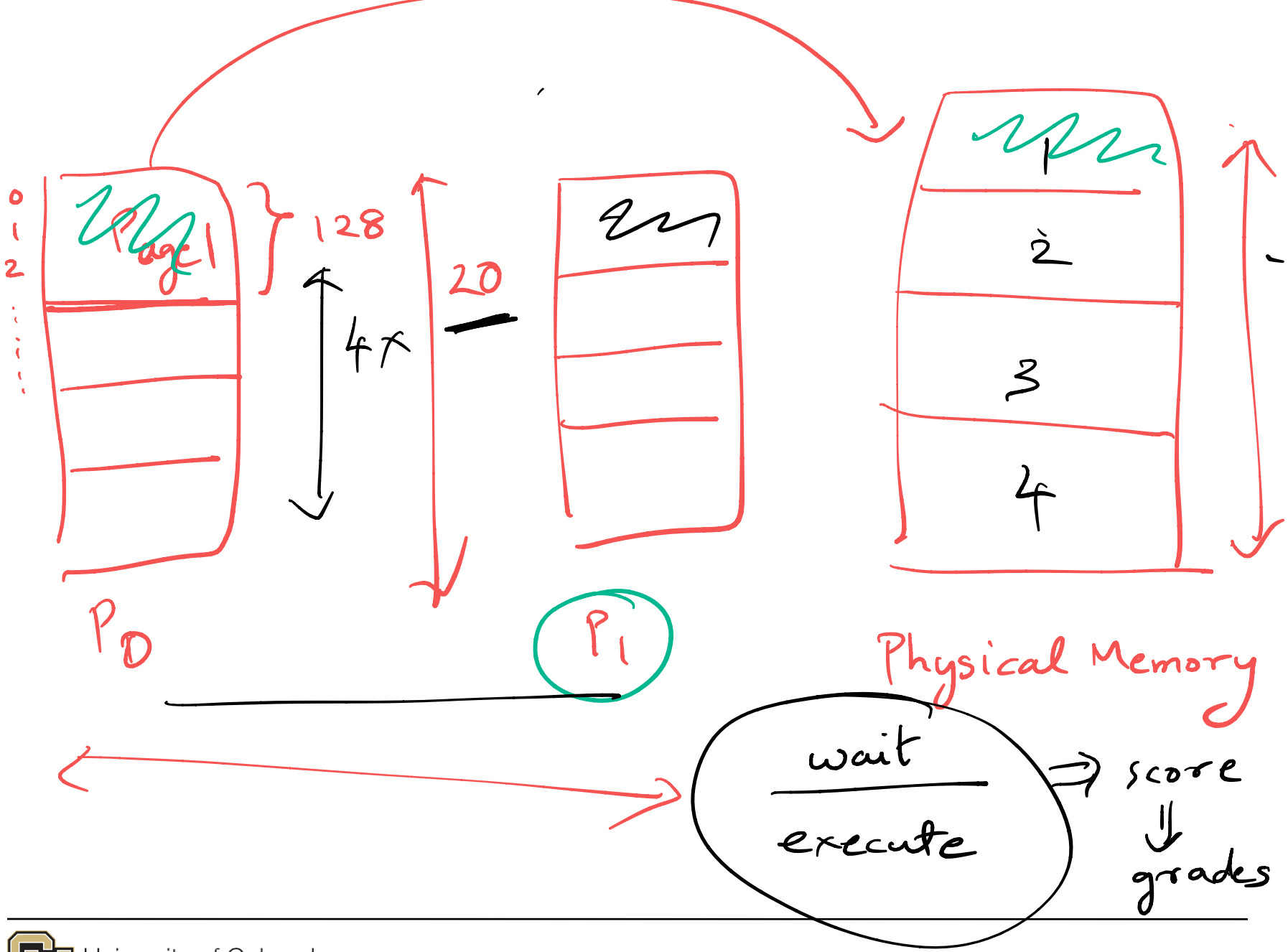
Assignment Goal

Implement a paging strategy that a paging simulator can use to maximize the performance of the memory access in a set of pre-defined programs

Action items

- Implement **LRU** algorithm
  `pager-lru.c`
- Implement any form of **predictive paging algorithm**
  `pager-predict.c`

15


0
1
2
page 1
128
4x
20
1
2
3
4
P_0
P_1
Physical Memory
wait
execute
score
grades

Paging Simulator

- Run a random set of 5 pre-defined programs utilizing a limited number of shared physical pages

- Provided default values in `simulator.h`
  - 20 virtual pages per process (MAX_PROC_PAGES)
  - 100 physical pages (frames) in total (PHYSICAL_PAGES)
  - 20 simultaneous processes competing for pages (MAX_PROCESSES)
  - 128 memory unit page size (PAGE_SIZE)
  - 100 tick delay to swap a page in or out (PAGE_WAIT)
    - Each instruction or step in the simulated programs requires 1 tick to complete.

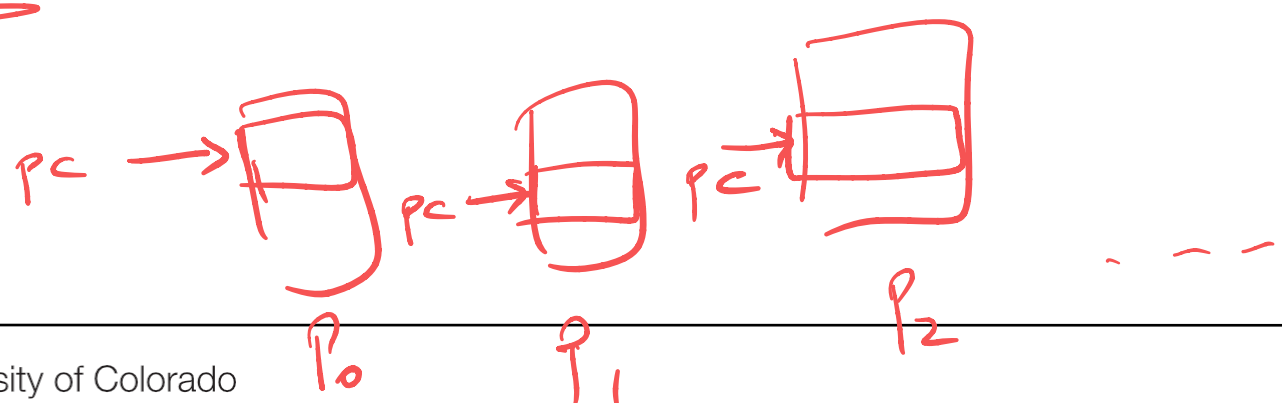
Paging Simulator

YES 100 pages

20 × 20 = 400

process page

- Is the environment resource constrained?
- How many physical pages can be swapped in at a given time?
- How many virtual pages will you have to access at most?
- Swapping a page in and out takes how much time?



Paging Simulator

process running?

which instruction running

```
struct pentry {
    long active;
    long pc;
    long npages;
    long pages[MAXPROCPAGES]; /* 0 if not allocated, 1 if allocated */
};
```

20

0 1 2

[1, 0 1, ...]

20

Paging Simulator

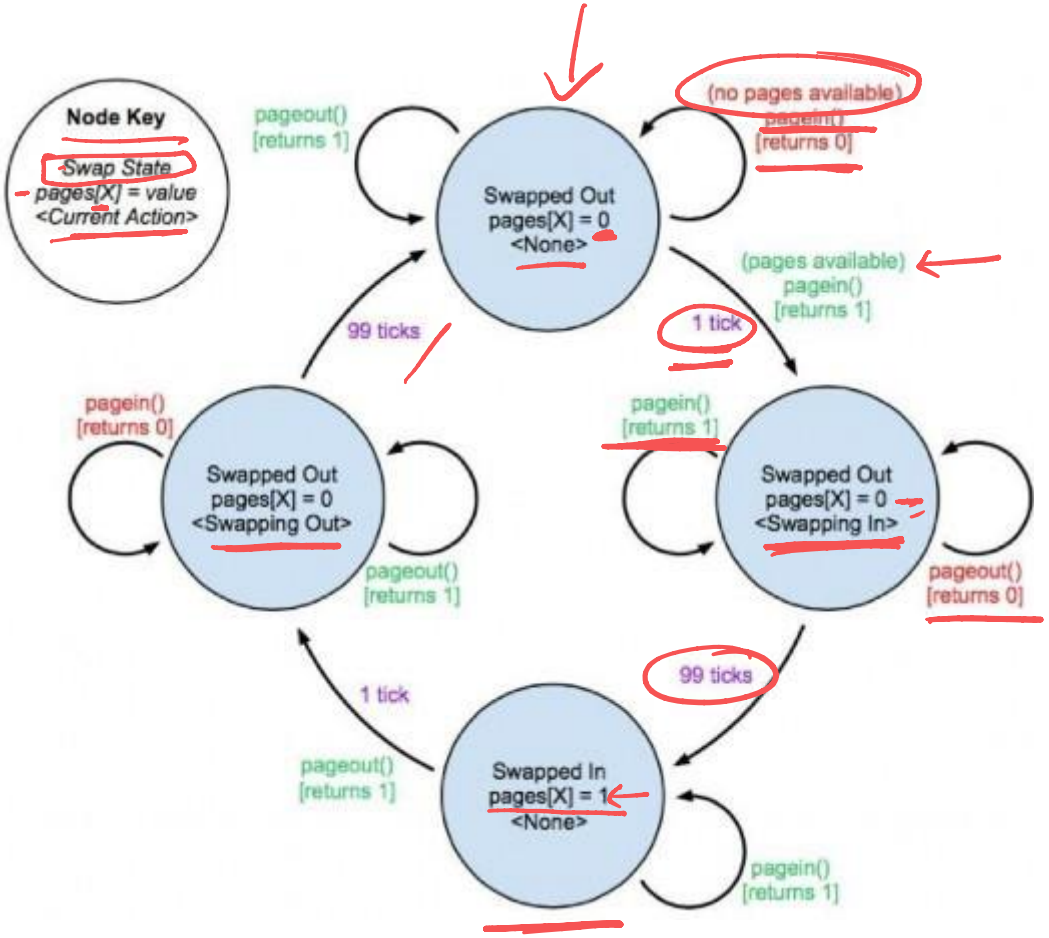
pagein(1, 20)

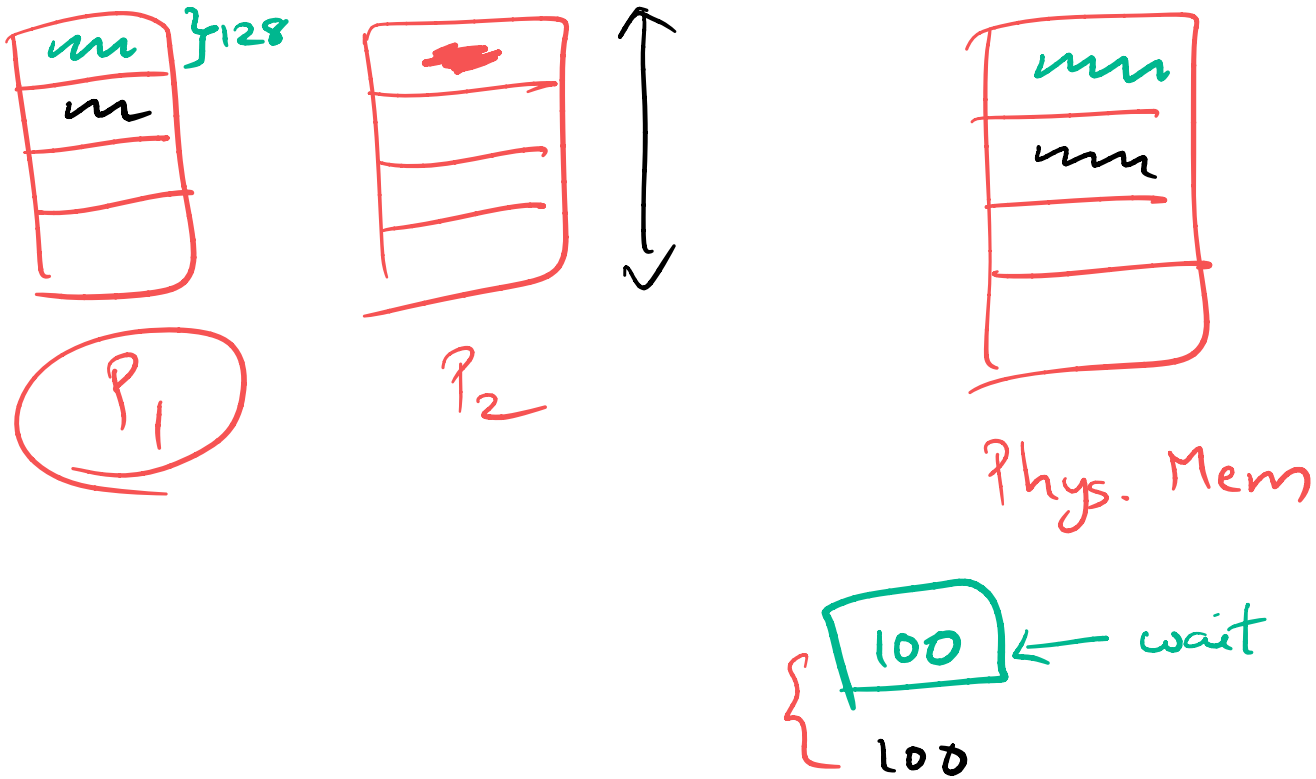
Provide 3 functions for interaction

- To control the allocation of virtual and physical pages
  - `pagein()`
  - `pageout()`
- To handle the page fault
  - **`pageit()`** ← core paging function that needs implementation

Source code is provided.

- `simulator.h, simulator.c`
- `pager-basic.c, pager-lru.c, pager-predict.c`


Node Key
Swap State
pages[X] = value
<Current Action>
pageout()
[returns 1]
Swapped Out
pages[X] = 0
<None>
(no pages available)
pagein()
[returns 0]
(pages available)
pagein()
[returns 1]
1 tick
99 ticks
pagein()
[returns 0]
Swapped Out
pages[X] = 0
<Swapping Out>
pageout()
[returns 1]
pagein()
[returns 1]
Swapped Out
pages[X] = 0
<Swapping In>
pageout()
[returns 0]
99 ticks
1 tick
pageout()
[returns 1]
Swapped In
pages[X] = 1
<None>
pagein()
[returns 1]


128
P1
P2
Phys. Mem
100
wait
100

Paging Simulator

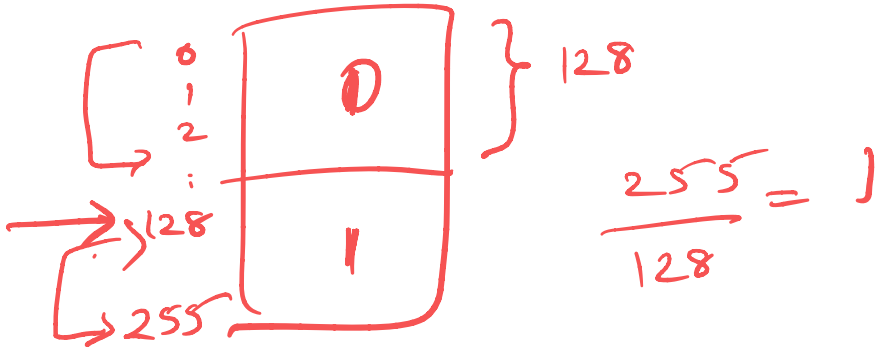
```
while (!alldone()) { // all processes inactive
    allstep();    // advance time one tick; if process done, reload
    allage();     // advance time for page wait variables.
    callyou();    // call your program
    sysclock++;     // remember new time.
    allblocked();   // deadlock detection
}
allscore();
```

pageit()

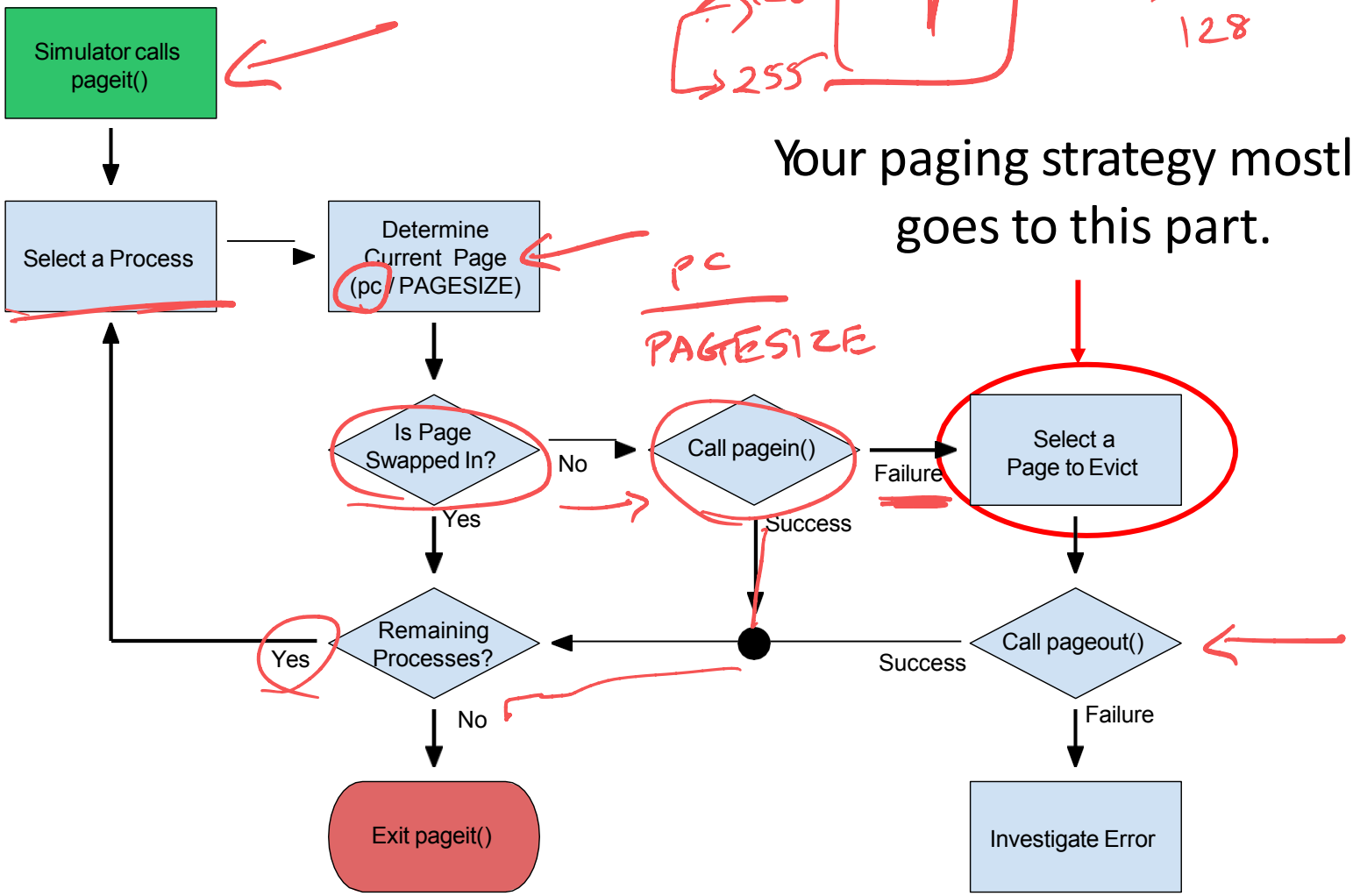
pager-basic.c

- A basic "one-process-at-a-time" implementation
- A simple demonstration of the simulator API
- Doesn't need any implementation from YOU!!!

pager-basic.c

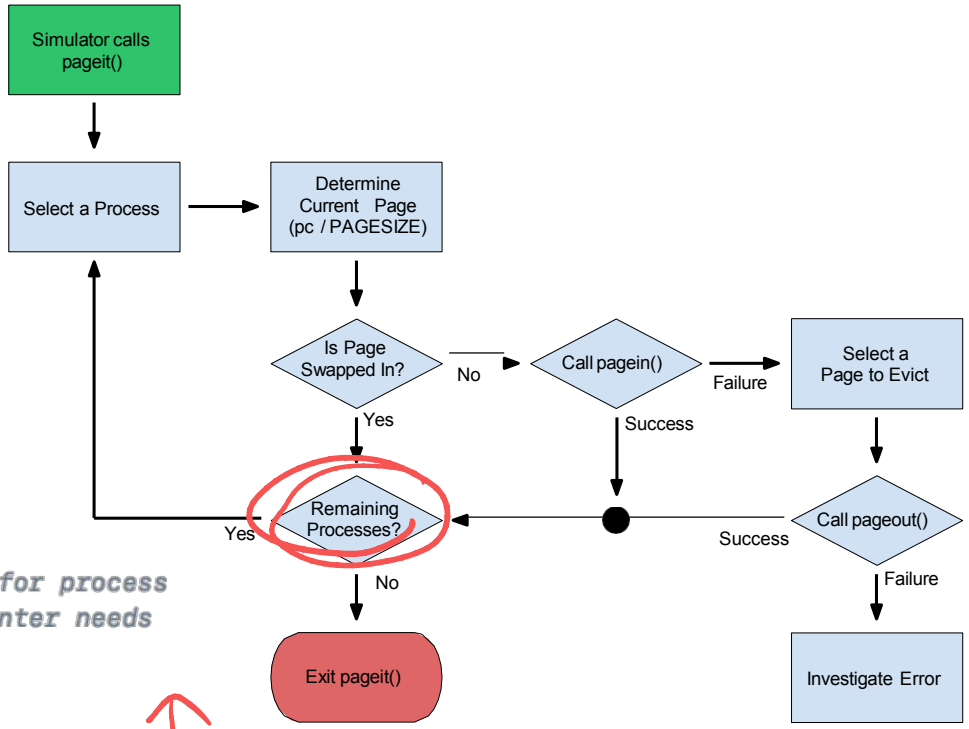


Your paging strategy mostly goes to this part.



```c
#include "simulator.h"

void pageit(Pentry q[MAXPROCESSES]) {

    /* Local vars */
    int proc;
    int pc;
    int page;
    int oldpage;

    /* Trivial paging strategy */
    /* Select first active process */
    for(proc=0; proc<MAXPROCESSES; proc++) {
    /* Is process active? */
    if(q[proc].active) {
        /* Dedicate all work to first active process*/
        pc = q[proc].pc;                // program counter for process
        page = pc/PAGESIZE;          // page the program counter needs
        /* Is page swaped-out? */
        if(!q[proc].pages[page]) {
        /* Try to swap in */
        if(!pagein(proc,page)) {
            /* If swapping fails, swap out another page */
            for(oldpage=0; oldpage < q[proc].npages; oldpage++) {
            /* Make sure page isn't one I want */
            if(oldpage != page) {
                /* Try to swap-out */
                if(pageout(proc,oldpage)) {
                /* Break loop once swap-out starts*/
                break;
                }
            }
            }
        }
        }
        /* Break loop after finding first active process */
        break;
    }
    }
}
```

pagein( )



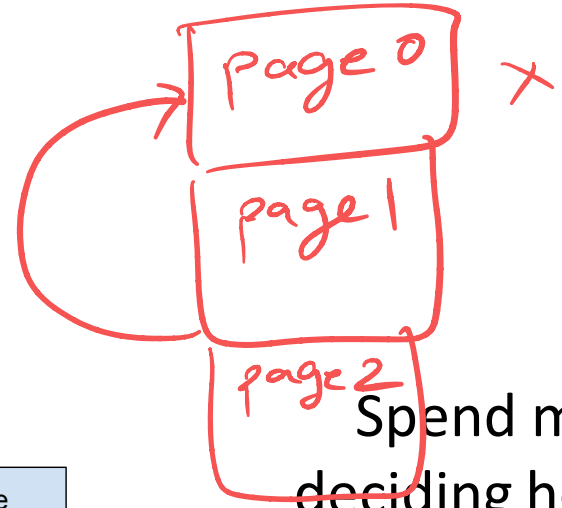
! pagein → 1

! 0

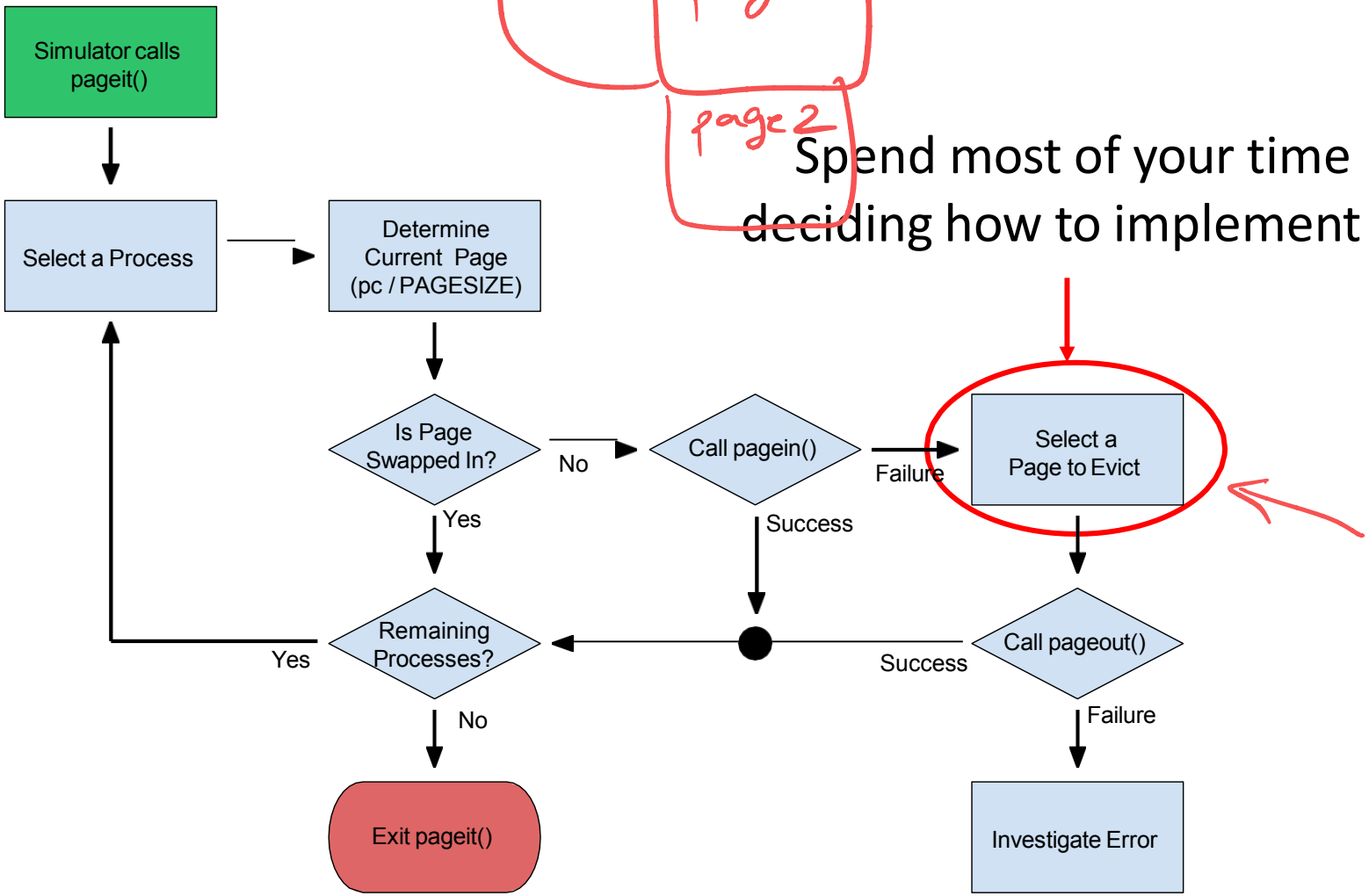
1

pager-basic.c

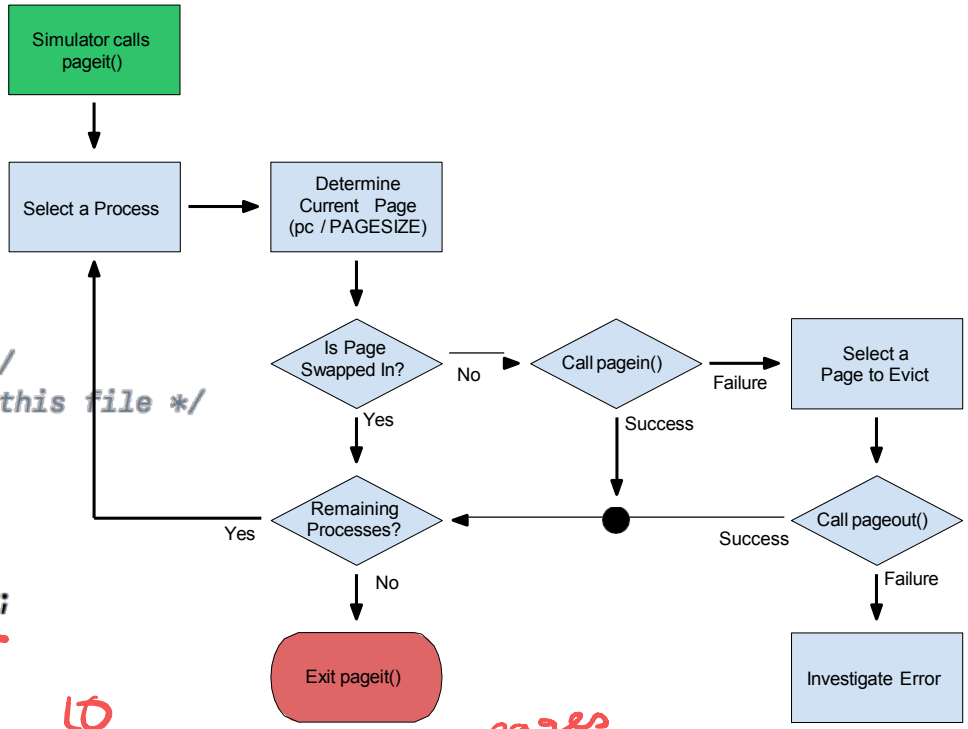
pager-lru.c



Spend most of your time deciding how to implement it



```c
#include <stdio.h>
#include <stdlib.h>

#include "simulator.h"

void pageit(Pentry q[MAXPROCESSES]) {

    /* This file contains the stub for an LRU pager */
    /* You may need to add/remove/modify any part of this file */

    /* Static vars */
    static int initialized = 0;
    static int tick = 1; // artificial time
    static int timestamps[MAXPROCESSES][MAXPROCPAGES];

    /* Local vars */
    int proctmp;
    int pagetmp;

    /* initialize static vars on first run */
    if(!initialized){
    for(proctmp=0; proctmp < MAXPROCESSES; proctmp++){
        for(pagetmp=0; pagetmp < MAXPROCPAGES; pagetmp++){
        timestamps[proctmp][pagetmp] = 0;
        }
    }
    initialized = 1;
    }

    /* TODO: Implement LRU Paging */
    fprintf(stderr, "pager-lru not yet implemented. Exiting...\n");
    exit(EXIT_FAILURE);
    tick++;
    /* advance time for next pageit iteration */
```



10

pages

0 1 2 3 20

processes

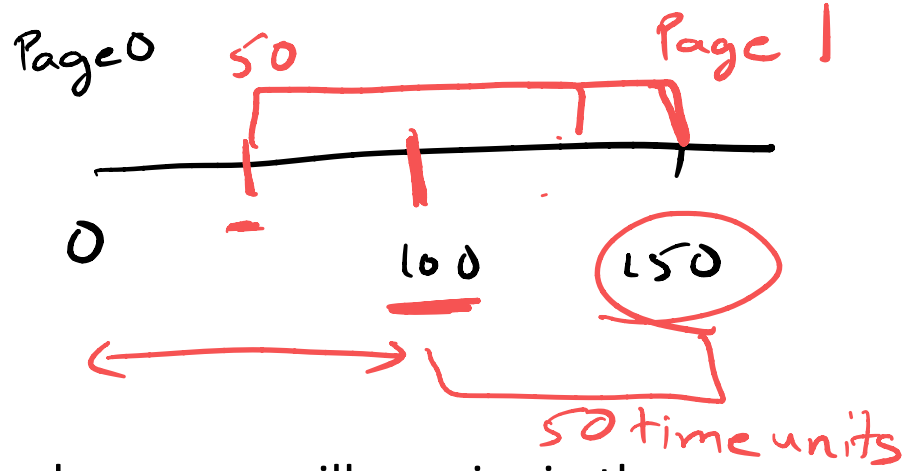
0

10 11 12 13

20

pager-lru.c

pager-predict.c

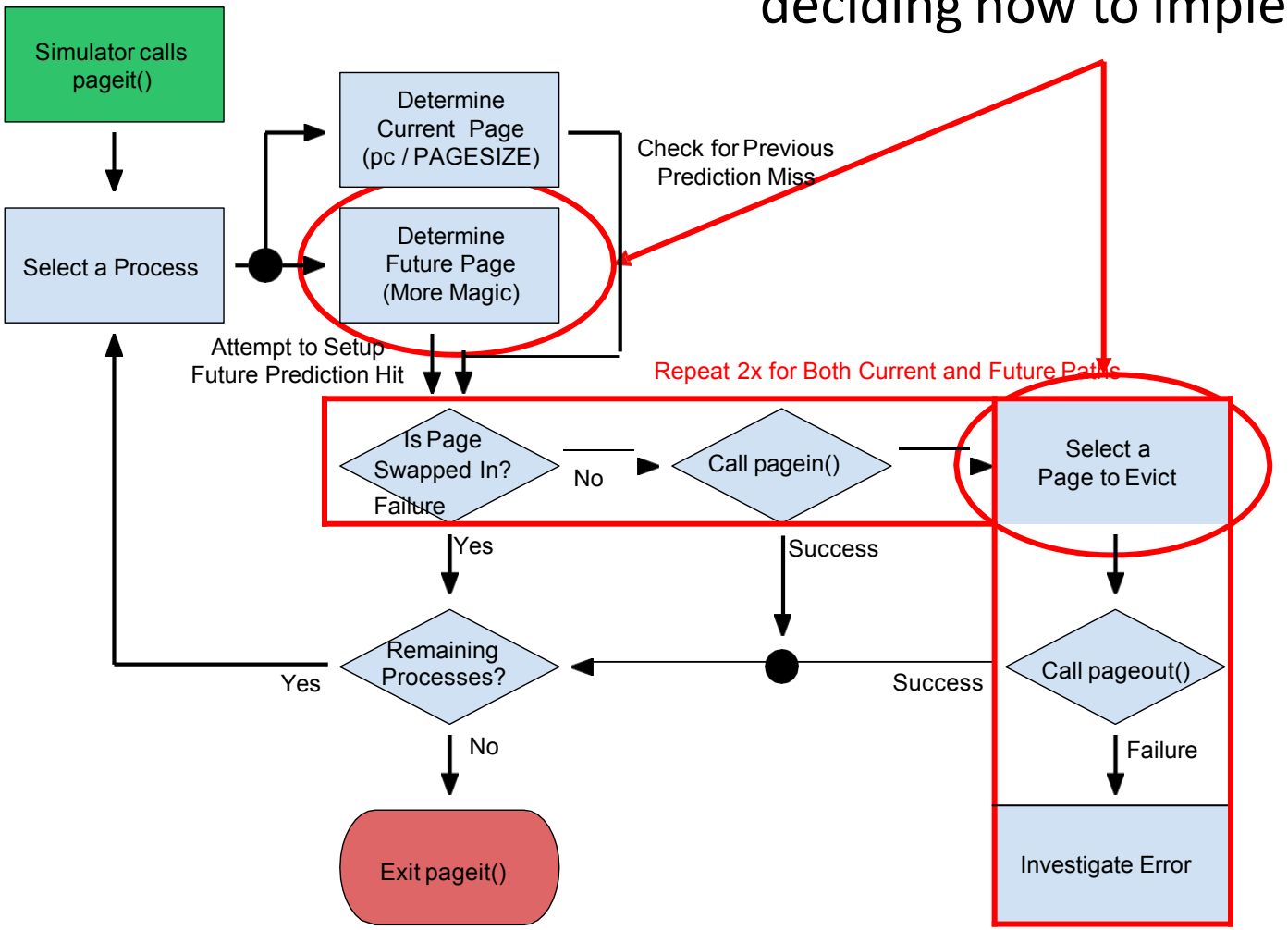


- Require a predictive algorithm that
  - Attempts to predict what pages each process will require in the future and then
  - Swaps these pages in before they are needed

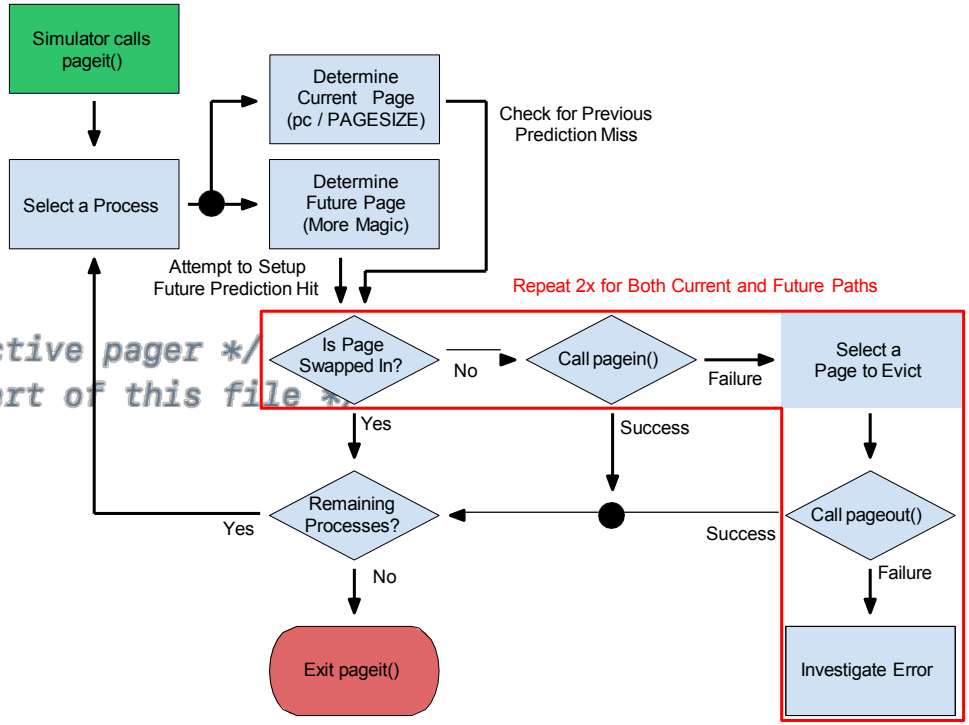
- **Note**: In any predictive operation, you ideally wish to stay 100-200 ticks ahead of the execution of each process.

pager-predict.c

Spend most of your time deciding how to implement it



```c
#include <stdio.h>
#include <stdlib.h>

#include "simulator.h"

void pageit(Pentry q[MAXPROCESSES]) {

    /* This file contains the stub for a predictive pager */
    /* You may need to add/remove/modify any part of this file */

    /* Static vars */
    static int initialized = 0;
    static int tick = 1; // artificial time

    /* Local vars */


    /* initialize static vars on first run */
    if(!initialized){
    /* Init complex static vars here */

    initialized = 1;
    }

    /* TODO: Implement Predictive Paging */
    fprintf(stderr, "pager-predict not yet implemented. Exiting...\n");
    exit(EXIT_FAILURE);

    /* advance time for next pageit iteration */
    tick++;
}
```



pager-predict.c

Week 11 – Checklist

- ❑ Start on PA4
- ❑ Complete Quiz 13