



# **CSCI-3753: Operating Systems Fall 2020**

**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`



# 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

- Is the environment resource constrained? ✓

20 process  $\times$  20 virtual pages = 400  
100 physical pages

- 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

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

running?

20

pages[20]



tick → pc  
tick += 1

## Paging Simulator

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

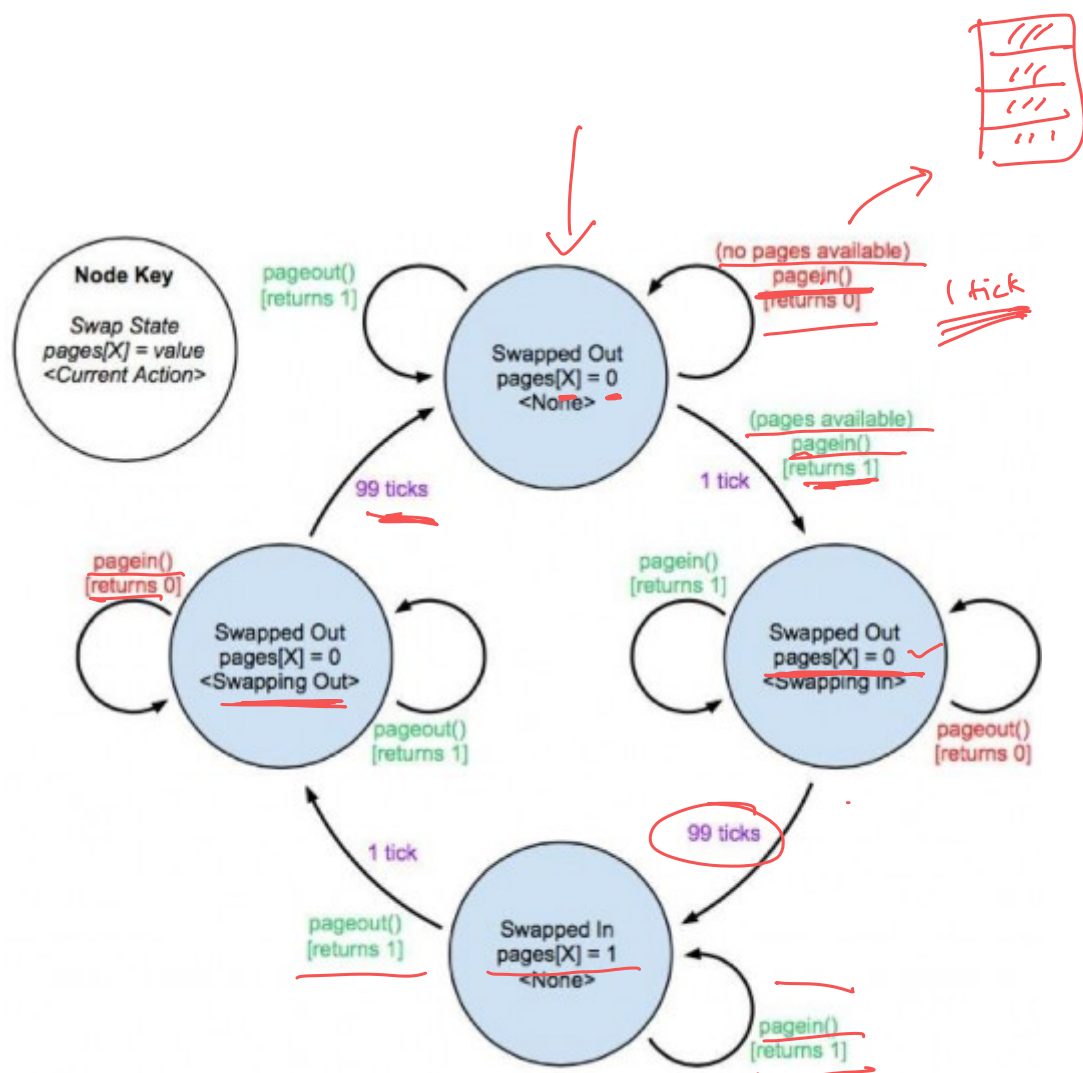
Source code is provided.

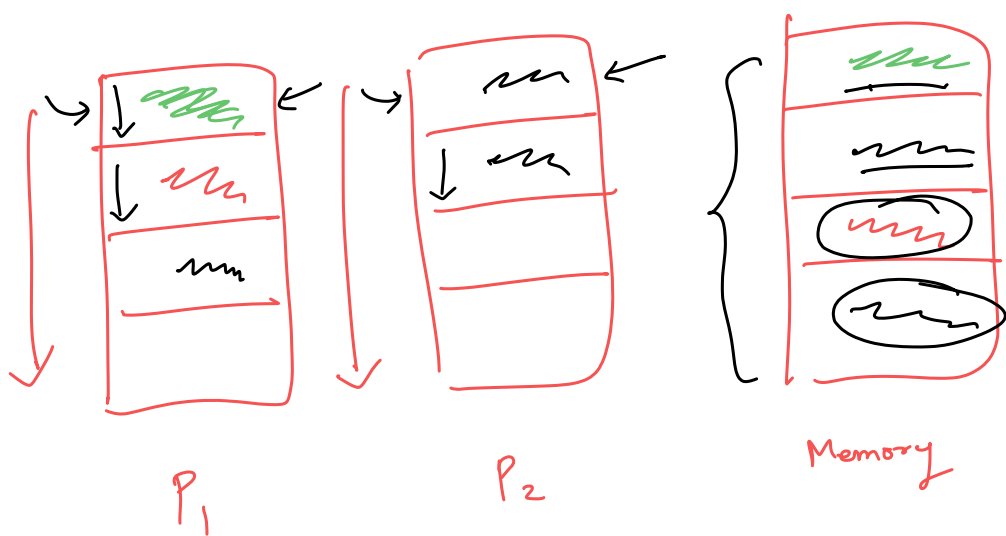
→ • simulator.h, simulator.c

→ • pager-basic.c, pager-lru.c, pager-predict.c





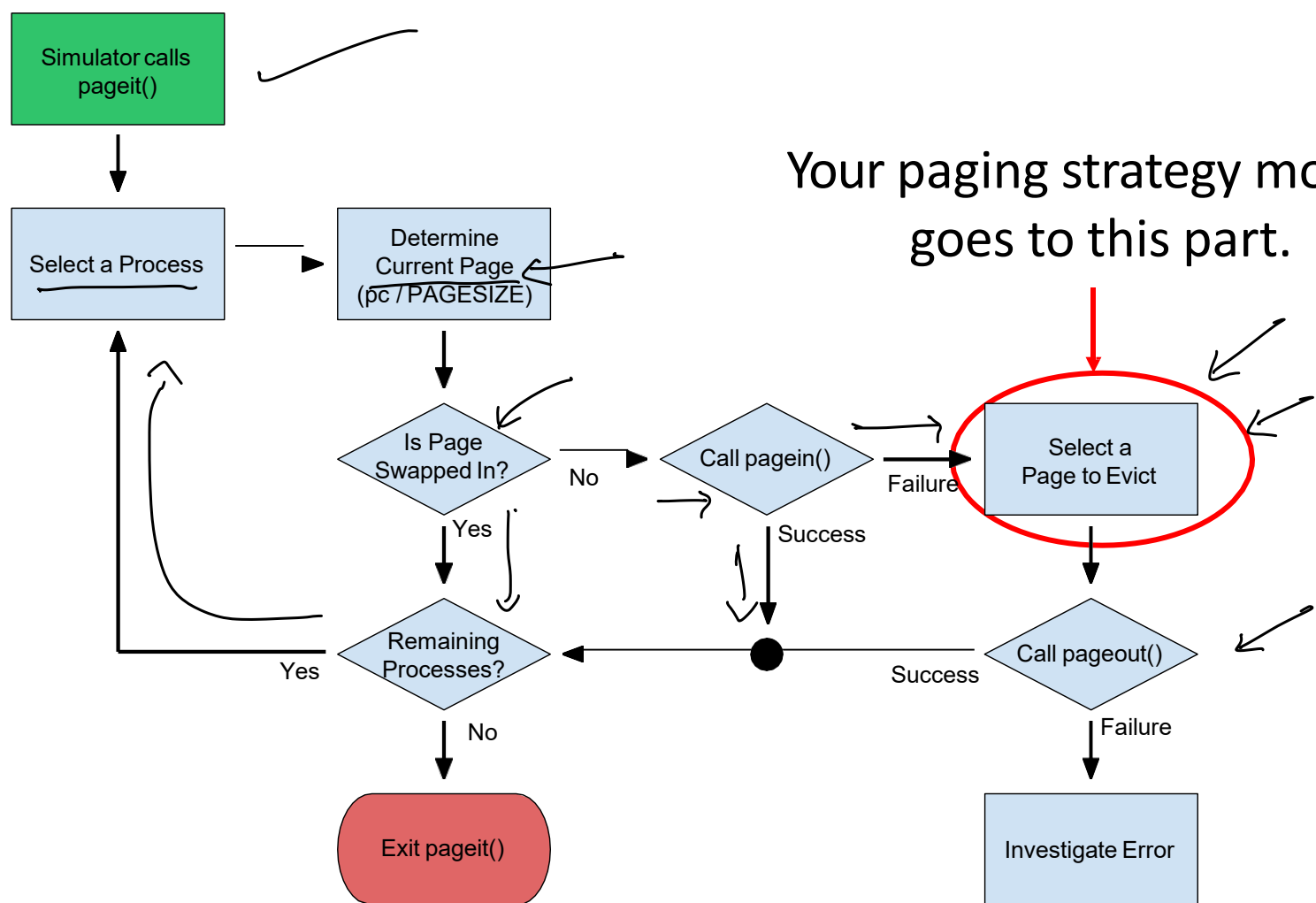
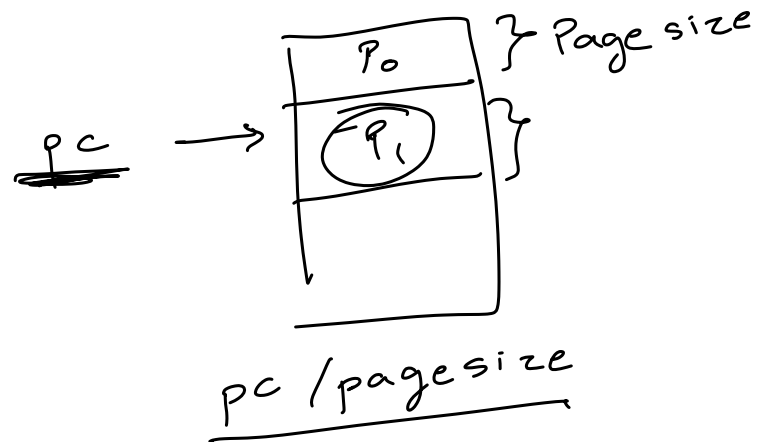




## pager-basic.c

- 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



```
#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;
```

```
            page = pc/PAGESIZE;
```

```
            /* Is page swapped-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;
```

```
                            }
```

```
                        }
```

```
                    }
```

```
            }
```

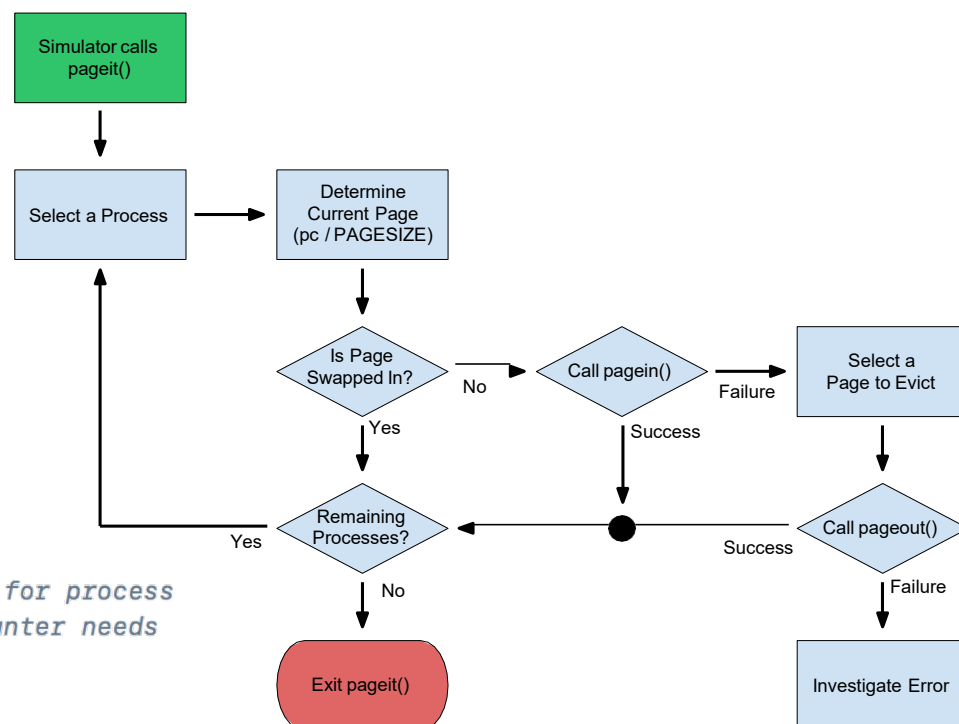
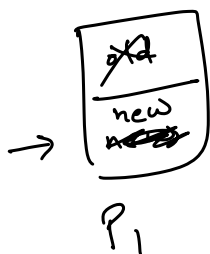
```
        }
```

```
        /* Break loop after finding first active process */
```

```
        break;
```

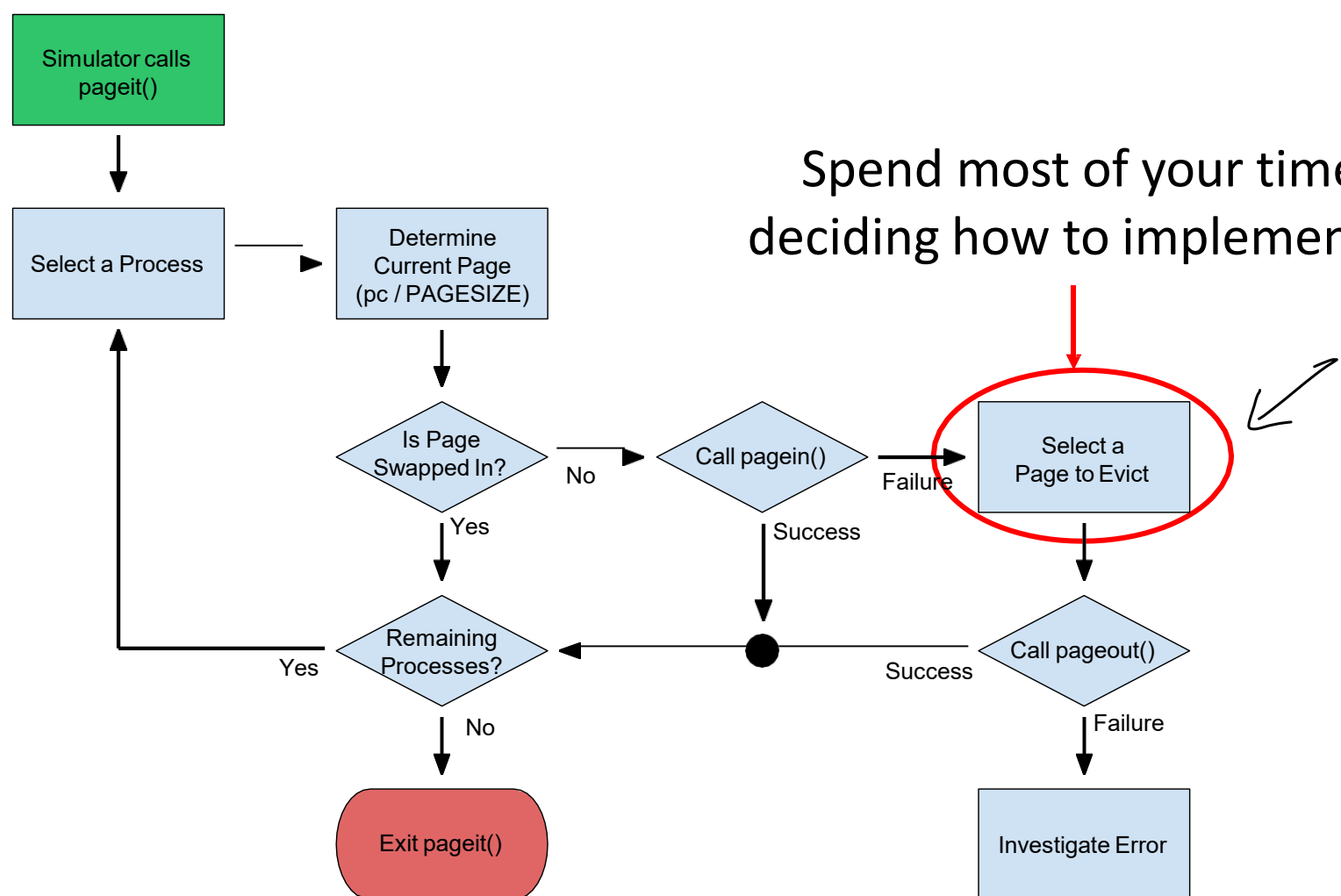
```
    }
```

```
}
```



page-r-basic.c

# pager-lru.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;
```

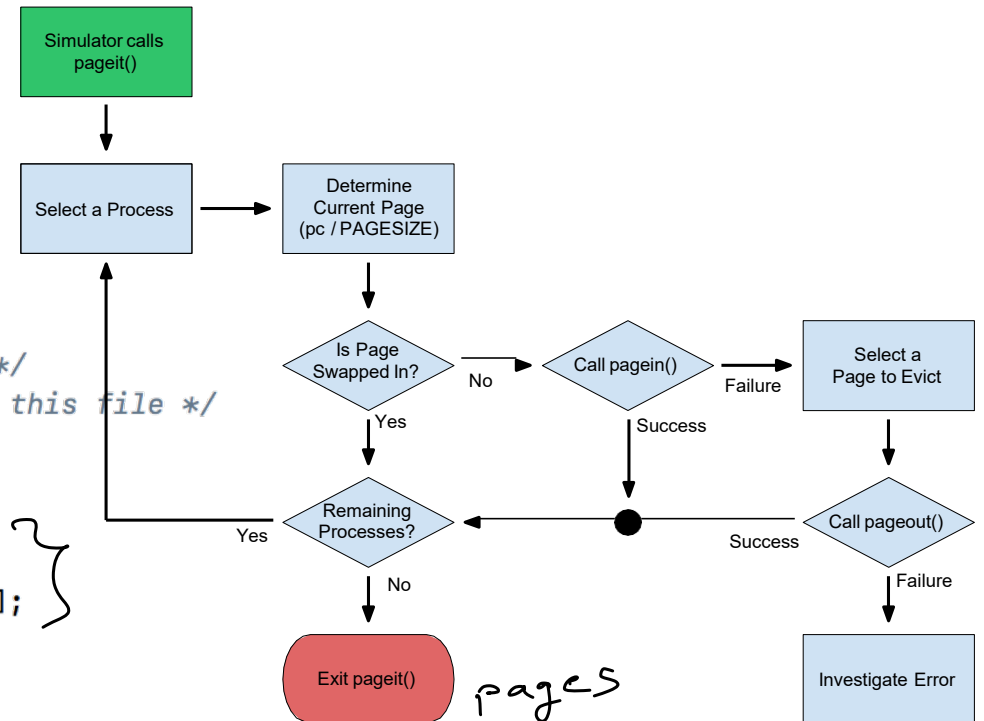
```
/* initialize static vars on first run */
```

```
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);
```

```
/* advance time for next pageit iteration */
```



pages

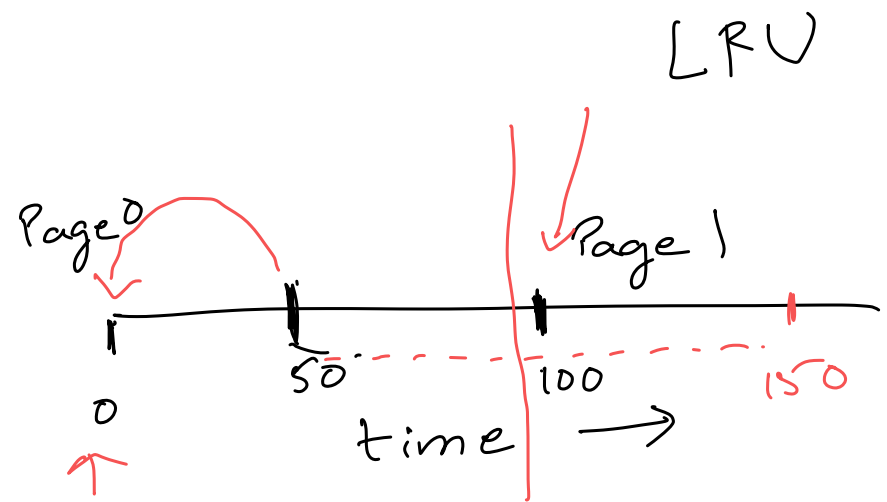
0 - ... 20

proc 0  
:  
:  
20

[ 1 - 2 3 4 5 ]

CSCI 353, Fall 12, 01.9

page r-lru.c



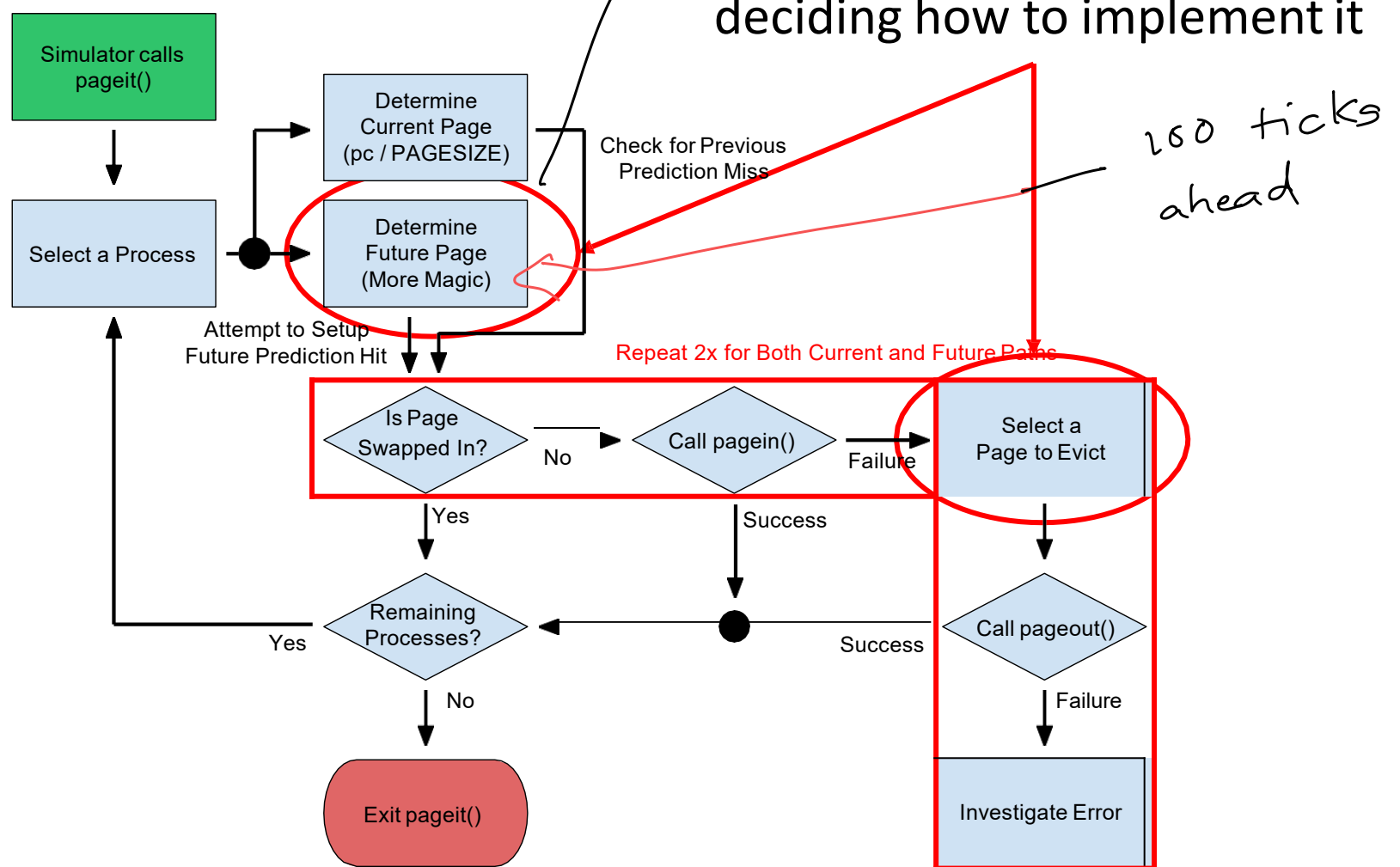
## pager-predict.c

- 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



```

#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 */

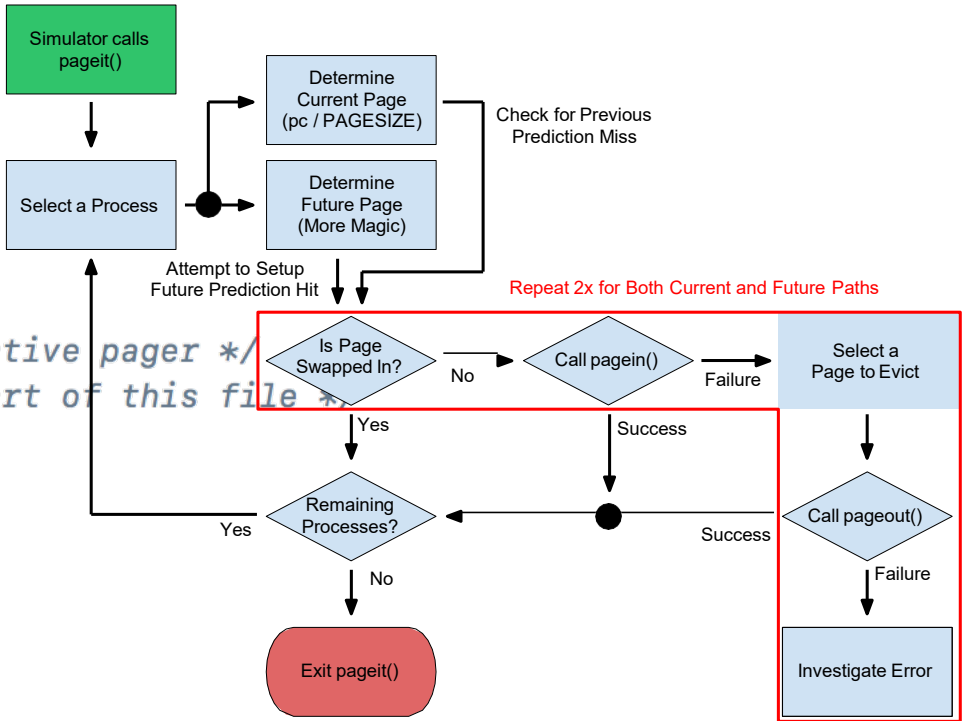
    /* initialize static vars on first run */
    if(!initialized){
        /* Init complex static vars here */

        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++;
}

```



## Week 11 – Checklist

- ☐ Start on PA4
- ☐ Complete Quiz 12  
PS3

