ECS150 Assignment 2 Hint

May 2022

Q3

Working Set

In paging replacement policy like FIFO, LRU:

- We do not need as many pages as a process need to reduce page faults.

There is, for any given computation, at any given time, a number of pages such that:

- if we increase the number of page frames allocated to that process, it makes very little difference in the performance.
- if we reduce the number of page frames allocated to that process, the performance suffers noticeably.

We will call that number the process' **working set size**.

Working Set Replacement

We need to predict appropiate size of pages each process has to use.

- The goals are first to specify which pages a given process needs to have memory resident in order for the process to run without too many page faults and second to ensure that these pages are indeed resident.
- As a proccess is running on OS, OS needs to make sure the process has enough pages for it to have minimum page faults.

Q4

- Each process has **one** of glue, newspaper, and model kits.
- The agent has **two** of the items for the model builder.

Which means,

- The agent needs to **signal** the missing item. For example, the agent has glue and newspaper, the agent should signal to the process which has model kits.
- Each process(or model builder) should **wait** for the agent to signal the specific model that the process has.

Code

```
/*Basic structure of the psuedocode, you need to explain every step you take.*/


typedef enum item = {glue, newspaper, model};
//handling agent's signal and the process' wait
monitor{
    /*There are some condition variables you need to set up for monitor*/

    //agent put two items to wait a process to take, you need to use the wait and signal
    entry void needed(item i){

    }

    //a builder process gave its item to agent to build an airplane, you need to use the wait and signal
    entry void take(item i){

    }
} build;

//builder process, it takes the missing item, builds and finish
void builder(item item_the_builder_miss){

}

//agent process
void agent(void){

}

//main function
```

```
31 void main(void){
32
33 }
```

Listing 1: coding structure

Extra 1

Belady's anomaly

- In computer storage, Bélády's anomaly is the phenomenon in which increasing the number of page frames results in an increase in the number of page faults for certain memory access patterns. This phenomenon is commonly experienced when using the first-in first-out (FIFO) page replacement algorithm. In FIFO, the page fault may or may not increase as the page frames increase, but in optimal and stack-based algorithms like LRU, as the page frames increase, the page fault decreases.
- The reason that FIFO has more page faults is that recently request pages can remain at the bottom of the FIFO queue longer.

For Optimal algorithm or stack-based algorithm like LRU, has property $M(n) \subseteq M(n+1)$, however, Bélády's anomaly does not. Try to prove $M(n) \nsubseteq M(n+1)$.