ICS Homework 13

June 6, 2020

1 Organization

1.1

Let's consider about a page-removal algorithm: clock algorithm. Suppose we have a primary device which has 3 physical blocks, every time a reference string P come, it will follow the pseudo-code:

```
if hit(P)
    res_block = block contains P
    referenced_bit[res_block] <- True
    clock_arm does not change
else
    while referenced_bit[clock_arm]
        referenced_bit[clock_arm] <- False
        clock_arm <- next block
    res_block = block[clock_arm]
    referenced_bit[clock_arm] <- True
    clock_arm <- next block
return res_block
```

Fill in the table (If you don't know what to fill just write down a '-'). Note: '*' means the position of the clock arm; you also need to tell what the referenced bit is for each page block at that time (1 for True and 0 for False).

Time	0	1	2	3	4	5	6	7	8
Reference string	-	3	4	2	6	4	3	7	4
Primary Device Contents	* - -	3 * -	3 4 *	3 * 4 2					
Referenced Bit	0 0 0	1 0 0	1 1 0	1 1 1					
Page Absent	-	Y	Y	Y					

1.2

Please consult the related information. Find and complement the definition of struct vm_area_struct in Linux v5.7.

```
struct vm_area_struct {
        /* Please complete me */
} __randomize_layout;
```

2 System Software

2.1

Let p denote the number of producers, c the number of consumers, and n the buffer size in units of items. Consider the following buffer implementation. For each of the following scenarios, indicate whether the **mutex** semaphore is necessary or not to implement function **sbuf_insert** and **sbuf_remove**.

```
typedef struct {
  int *buf;      /* Buffer array */
  int n;         /* Maximum number of slots */
  int front;     /* buf[(front+1)%n] is first item */
  int rear;      /* buf[rear%n] is last item */
  sem_t mutex;   /* Protects accesses to buf */
  sem_t slots;   /* Counts available slots */
  sem_t items;   /* Counts available items */
} sbuf_t;
```

A. $p = 1, c = 1, n > 1$
B. $p = 1, c = 1, n = 1$
C. $p > 1, c > 1, n = 1$