Alan Palayil CS 450 Homework 2

Explain the concept of Copy on Write (CoW). Explain how Copy on Write works when a child process is forked from its parent. Why is it more efficient and saves memory?

- A technique to copy the data resources into a computer system on write function, the copy can be existed as a reference to the true or original data when the data is only copied but not modified. So, from here we can say that only modified data is copied data. When a child process is forked from its parent the main use of the copy on write is to sharing the virtual memory address of the OS processes but in the implementation of the fork system calls. The process never modify memory, but it immediately executes a new and fresh memory by just exchanging the address space with it entirely address. It is more efficient because this technique can be extended to support the maximum efficient memory allocation and it also save memory because of its extended nature we can use the copy on write for more works etc.

A group of 5 people goes to a restaurant. They wait until the last person arrives before they start ordering. Implement this scenario using threads and semaphores. (Hint: use pthread_create, sem_init, sem_wait, sem_post.)

```
sem_t s [5]; //initialize each s[i] to 0

void person (int p) //p is set to be between 0 and 4

{

        int i; printf (“Person %d has arrived.\n”, p+1);

        for (i=0; i<4; i++)

                sem_post(s[p]); //signal other people that this person has arrived.

        for (i=0; i<5; i++)

        {

                if (i! = p)

                        sem_wait(s[i]); //wait for other people

        }
```

```
        printf ("Person %d starts ordering.\n", p+1);
}
```

Compare spinlock and semaphore. Both can be used to protect a critical region. In what case is spinlock better? In what case is semaphore better?

- Spinlock is good for short critical regions with no I/O operations in it. Semaphore is good for long critical regions which may have I/O operations.

Consider a 32-bit address space with 4KB pages. Assume each PTE is 4 bytes:

How many bits do we need to represent the offset within a page?

- 12

How many virtual pages will we have?

- 2^{20}

What will be the overall size of the page table?

- 4MB