Ahmedabad University
OS Lab exam descriptions

Enrollment no: AU1940207 Name: Keyur Nagar

Answer 1:

in this question, we are supposed to print the pattern which looks as follow:
A 1 a B 2 b C 3 c ....Y 25 y Z 26 z

we will use multithreading method to print the pattern. If we observe the pattern carefully, one can find that there will be 3 letters which repeat again and again(e.g capital alphabets-number-small alphabets)

we can convert the capital and small alphabets to ascii and can print them accordingly. For that we will use small equations and will use loop and iterate through and calculate the equation and find the corresponding value.

The equation of the ascii value given below:

capital=i+65
small=i+97
number=i+1

where i = 0,1,2,.....,25

Answer 2B:

Buddy system:

Buddy system is similar to fixed partition. For implementing Buddy’s system we use Buddy’s algorithm. In this algorithm, entire memory is treated as single block which is power of 2(e.g 2^U).

if the address $2^{U-1}<S<=2^U$ We allocate the given block of the memory to process or page.

Else we recursively divide the block equally and test the above condition again and again until it reach them.

For example:

Consider the memory of size 1 MB

So available space will be

1MB

if the request S =244K comes then we need 0 to 256K size of location for storing 244K

hence, we need to divide above 1MB space recursively.

256K	**256K**	512K

note that the BOLD underlined number allocated to some process.

If request S=100 comes then 64<100<=128 will be divided recursively and 0 to 128 allocated to the process

128K	128K	**256K**	512K

Same way , if request for 63K comes then above 128K unallocated memory subdivided into 64K of memory and one of them allocated to the 63K process.

128K	**64K**	64K	**256K**	512K

if the process having size 63K is completed or suspended then it will again goes to it previous state.

128K	128K	**256K**	512K

Same way if process having size 100K completed then it will deallocate as follow

256K	**256K**	512K

We use the exact same logic in implementing this algorithm.

Answer 3:

in producer-consumer problem, one or more than one producer produces the data and consumer taking the data out of the buffer. Only one of them can use the buffer or memory at a time. If producer produces consumer can’t consume and if consumer consumes producer can’t produce. If the buffer is full then producer can’t produce and if buffer is empty consumer can’t consume.

Solution using semaphores:

```
Void producer(){

        while(true){
                produce();
                Waitl(s);
                append();
                Signal(s);
                Signal(n);
        }
}

void Consumer(){
        while(true){
                Wait(n);
                Wait(s);
                consume();
                Signal(s);
                consume();

        }
}
```