

# CSCE611 Machine Problem 2 Frame Manager

Gehao Yu 629008717

Feb 11 2021

## 1 Introduction

We have to design a frame manager based on the requirement of the machine problem 2. The total memory space in the computer is 32MB, while the kernel space is from 0 to 4MB, and user space from 4MB to 32MB.

## 2 Initialization Process

Once we need to initialize a frame pool, we would create a bitmap and assign its length as 4 times of pool's frame size. Because every two bits in the bitmap would represent the status of one frame in the current frame pool. Then we initialize all the frame as status "Free" and if the info frame number is the base frame, we mark it as used. ("11" represented Free, "01" represented Used).

```
//directly give address to the 1st position of bitmap
if(info_frame_no == 0) {
    bitmap = (unsigned char*) (base_frame_no * FRAME_SIZE);
    isHead = (unsigned char*) (base_frame_no * FRAME_SIZE);
}else{
    bitmap = (unsigned char*) (info_frame_no * FRAME_SIZE);
}

//initially, all frames are unallocated, not the head of sequence
for(int i = 0; i * 4 < nframes; i++){
    bitmap[i] = 0xFF;
}

//When the info frame number is the base frame, then we mark
//that frame as used
if(info_frame_no == 0){
    bitmap[0] = 0x7F;
    nfreeframes--;
}
```

### 3 Helper Function-check\_state

Thanks to the hint from the professor, we implemented a helper function called "check\_state". The input variables are target frame number and status check mask. Once we get these two variables, we minus the target frame number with base frame number and divide with 4, then we got the bitmap index of target frame. Then we use the remainder of the difference divide 4 as the offset. Using the bit-wise AND, then we could get the current status of target frame number.

```
unsigned int ContFramePool::check_state(unsigned long _target_frame_no, unsigned int _input_mask){
    int frame_diff = _target_frame_no - base_frame_no;
    unsigned int bitmap_index = frame_diff / 4;
    unsigned char mask1 = _input_mask >> (frame_diff % 4);
    if((bitmap[bitmap_index] & mask1) == 0){
        return 1; //Used
    }else{
        return 0; //Free
    }
}
```

### 4 Helper Function-set\_state

Thanks to the hint from the professor, we implemented another helper function called "set\_state". The input variables are target frame number and status check mask. Once we get these two variables, we minus the target frame number with base frame number and divide with 4, then we got the bitmap index of target frame. Then we use the remainder of the difference divide 4 as the offset. Using the bit-wise XOR, then we could set the current status of target frame number as we wish.

```
void ContFramePool::set_state(unsigned long _target_frame_no, unsigned int _input_mask){
    unsigned int index = (_target_frame_no - base_frame_no) / 4;
    unsigned char mask = _input_mask >> (_target_frame_no - base_frame_no) % 4;
    bitmap[index] ^= mask;
}

unsigned long ContFramePool::get_frames(unsigned int _n_frames){
```

### 5 get\_frame

First we have to maintain three variables "cur", "distance" and "visited". Variable "cur" represented the index of traversing process, variable "distance" represented the position of last try process(a process that try to find required length of consecutive memory space) begun, and the variable "visited" represented

how long consecutive memory space last try process(a process that try to find required length of consecutive memory space) achieved.

Firstly, we would see if the last try process successfully achieved required length consecutive memory space. If not, we should keep traverse until we find a "Free" frame, and assign this distance to variable "distance". Then, we should keep traverse until we find a "used" frame or reached required length. If current position exceeded the maximum of the frame pool, return 0. Repeated this process. If we get the required space, then we should mark these position as inaccessible and filled the head bitmap position as used.

```

unsigned long ContFramePool::get_frames(unsigned int _n_frames){

    assert(nfreeframes >= _n_frames);
    int cur = 0;
    int distance;
    unsigned int visited=0;
    unsigned int cur_head = base_frame_no;
    while(visited<_n_frames){
        unsigned int frame_pos = cur+base_frame_no;
        unsigned int cur_state = check_state(frame_pos, 0x80);
        while((cur<nframes) && (cur_state != 0)){ //Not free
            cur ++;
            frame_pos ++;
            cur_state = check_state(frame_pos, 0x80);
        }
        distance = cur;
        while((cur<nframes) && (cur_state == 0)){
            cur ++;
            frame_pos ++;
            cur_state = check_state(frame_pos, 0x80);
            if(cur-distance==_n_frames){
                break;
            }
        }
        visited = cur-distance;
        if(cur>=nframes){
            return 0;
        }
    }
    cur_head += distance;
    mark_inaccessible(cur_head, _n_frames);
    unsigned char temp_mask2 = 0x08>>((cur_head-base_frame_no)%4);
    bitmap[cur / 4] ^= temp_mask2;
    return cur_head;
}

```

## 6 mark\_inaccessible

The input variables are base frame number and length of how many frames needed to be labeled as used. Once we get these two variables, we set all the required frames as used and minus same length of free frames, then quit.

```
void ContFramePool::mark_inaccessible(unsigned long _base_frame_no, unsigned long _n_frames)
{
    int i;
    for(i = _base_frame_no; i < _base_frame_no + _n_frames; i++){
        assert ((i >= base_frame_no) && (i < base_frame_no + nframes));
        set_state(i, 0x80);
    }
    nfreeframes -= _n_frames;
}
```

## 7 release\_frames

The input variable is the first frame number needed to be released. Once we get it, we traverse all the frame pools and once we find out that current frame is labeled as "used", we would set it as "free" till the end.

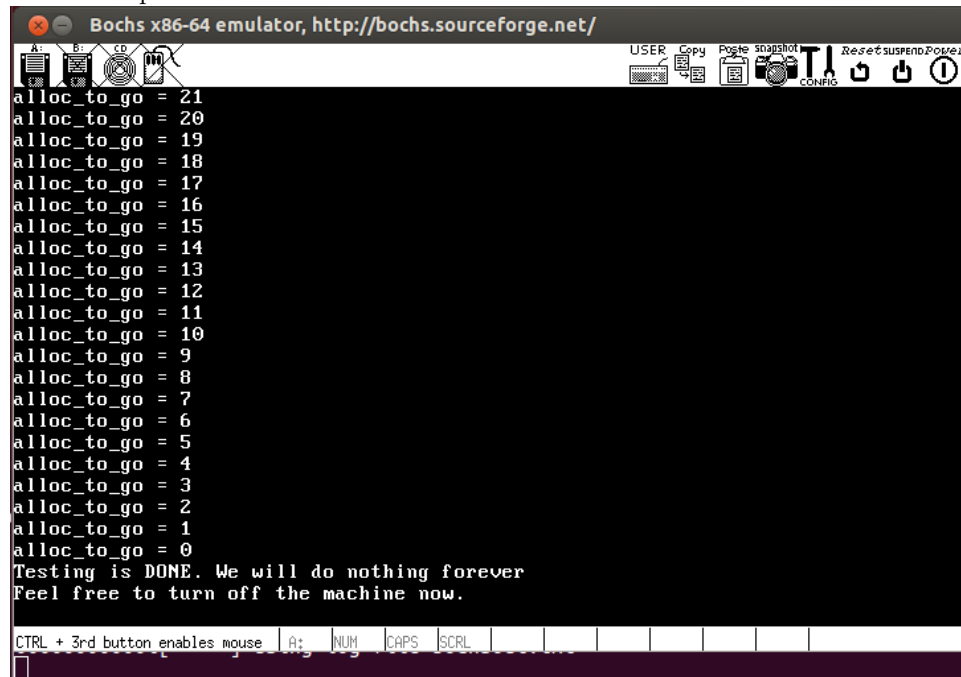
```
unsigned int cur_state = cur_pool.check_state(_first_frame_no, 0x08);

if(cur_state == 1){
    cur_pool.set_state(_first_frame_no, 0x08);
}

int i;
for(i = _first_frame_no; i < cur_pool.base_frame_no + cur_pool.nframes; i++)
    //when reach the head of sequence
    unsigned int head_state = cur_pool.check_state(i, 0x08);
    if(head_state==1) break;
    //when there are free frames
    unsigned int free_state = cur_pool.check_state(i, 0x80);
    //Free->0
    if(free_state==1) break;
    cur_pool.set_state(i, 0x80);
}
```

## 8 Result

The result performance of the code:



```
Bochs x86-64 emulator, http://bochs.sourceforge.net/
USER Copy Paste snapshot CONFIG Reset suspend Power
alloc_to_go = 21
alloc_to_go = 20
alloc_to_go = 19
alloc_to_go = 18
alloc_to_go = 17
alloc_to_go = 16
alloc_to_go = 15
alloc_to_go = 14
alloc_to_go = 13
alloc_to_go = 12
alloc_to_go = 11
alloc_to_go = 10
alloc_to_go = 9
alloc_to_go = 8
alloc_to_go = 7
alloc_to_go = 6
alloc_to_go = 5
alloc_to_go = 4
alloc_to_go = 3
alloc_to_go = 2
alloc_to_go = 1
alloc_to_go = 0
Testing is DONE. We will do nothing forever
Feel free to turn off the machine now.
CTRL + 3rd button enables mouse  A: NUM CAPS SCRL
```