

Design of MP4 – A formal Version

In previously submitted version, there are many problems that were not be solved. And now I submit a complete version again, in which all assignments in our handout have been completed and tested.

The first thing I updated successfully from MP3 is the contframepool, in which I made some modifications to decrease fragmentation and improve the use ration of our memory significantly. Attached in Figure 1 shows the modification. I just add a special condition to handle getting only one frame. Because my previous versions from MP1 to MP3 are only friendly to consecutive frames management. If I use them to get only one frame, the use ration of our memory will only be 25%.

```
unsigned int i = 0, hitting_target = 0;
if(_n_frames==1)
{
    unsigned int index;
    while(((i*4+_n_frames) <= nframes) && (hitting_target==0))
    {
        unsigned char searcher = 0xC0;
        for(unsigned int j = 0; j < 4; j++)
        {
            if((searcher&bitmap[i]) != 0)
                searcher = searcher>>2;
            else
            {
                hitting_target = 1;
                bitmap[i] = bitmap[i] | (0x40>>(2*j));
                index = j;
                break;
            }
        }
        i++;
    }
}
```

Figure 1. Modification in contframepool. C

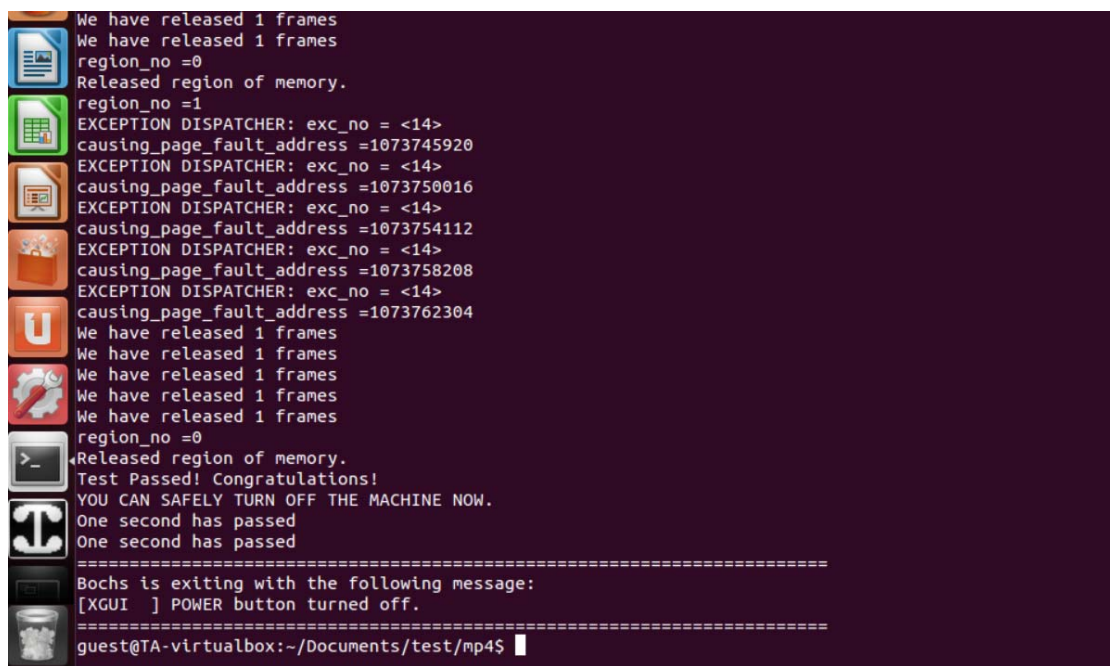
And then I faced the most important, also the most difficult part, the Recursive Page Table Look-up. After struggling for a long time and discussing with my classmates, I finally understand and I succeed in setting both page directory and page table pages in the process memory pool. In my program, I set two pointers, page_directory_entry and page_table_entry, pointing to the page directory and the page table page. And I calculate two indexes, PDE and PTE, of these two pointers. Thus, we can access the elements of the page directory and the page table page just like array reference, which are like that, page_directory_entry[PDE] and page_table_entry[PTE].

At the end of this project, I set VMPool. There are two important points in VMPool. The first one is how we allocate a new region and the second is how we free an existed region.

For allocating a new region, I divided it into several conditions. If there is no existed region, we can just set this region from (base_address + 4096), because the first page is occupied to record region information. And if there are several existed regions, we do search from the last interval, which is marked by the last existed region and the end of

this VMpool, to the first interval, which is marked by the second page ($\text{base_address} + 4096$) of this VMpool and the first existed region. Every time we find an interval to insert a new region, we would also insert a new element in the two-dimension array in the first page of this VMpool, the address of which is from base_address to $(\text{base_address} + 4096)$. We use `region_descriptors[1024]` to record allocated region, where elements with even number record starting address and elements with odd number record required pages. Actually, this array is one-dimension. However, we use it as two-dimension.

And for freeing an existed region, it is much easier. We can just invoke `PageTable::free_page` in a for circulation to free all pages in the region and delete two elements (the starting address and required pages of this region) in `region_descriptors[1024]`. Attached in Figure 2 shows the testing result.



```

We have released 1 frames
We have released 1 frames
region_no =0
Released region of memory.
region_no =1
EXCEPTION DISPATCHER: exc_no = <14>
causing_page_fault_address =1073745920
EXCEPTION DISPATCHER: exc_no = <14>
causing_page_fault_address =1073750016
EXCEPTION DISPATCHER: exc_no = <14>
causing_page_fault_address =1073754112
EXCEPTION DISPATCHER: exc_no = <14>
causing_page_fault_address =1073758208
EXCEPTION DISPATCHER: exc_no = <14>
causing_page_fault_address =1073762304
We have released 1 frames
We have released 1 frames
We have released 1 frames
We have released 1 frames
We have released 1 frames
region_no =0
Released region of memory.
Test Passed! Congratulations!
YOU CAN SAFELY TURN OFF THE MACHINE NOW.
One second has passed
One second has passed
=====
Bochs is exiting with the following message:
[XGUIS ] POWER button turned off.
=====
guest@TA-virtualbox:~/Documents/test/mp4$

```

Figure 2. Testing Result