Background

You need to add 2 shared memory pages (SMPs) to the address space of each process. The physical frame to which each SMP is mapped is the same for all processes, hence, “shared memory”.

The first page, the static SMP, is mapped to a physical frame when an address space is initialized. The second page, the dynamic SMP, is mapped to a physical frame when a process first writes to it.

1. Modifying the exec function to reserve 2 extra pages in the process address space

* There is a variable that represents the current size of the process’s memory. After the user stack is prepared, it points to the address **just after** the top of the user stack (top of page n+2).
* You need to add space for the 2 SMPs after the user stack.

1. Save the virtual address of the static SMP (page n+3) to the PCB
2. Increment the size of the process’s memory by the size of 1 page
3. Map this virtual page to the static SMP’s physical frame

* Repeat the above for the dynamic SMP. However, the physical frame of the dynamic SMP is mapped **when a process first writes to it.** So for step 3, the page should be mapped to an irrelevant physical frame as **read-only**. When a process tries to write to it, a **page fault** will be generated.

Hints:

* xv6 has a function for **mapping** virtual addresses to physical addresses.

2. Modifying fork to set up child process

fork creates a child process that is essentially a copy of the parent process. You need to update fork to copy over any new parts of the PCB that you have added.

To generate the child’s page table, fork walks through a copy of the parent’s page table, mapping the PTEs to **newly allocated** physical frames.

This is a problem for the SMPs. You’ll need to manually map the child’s SMP virtual addresses to their corresponding physical frames, like you did in exec.

However, there is a problem with using the xv6’s aforementioned **mapping function.** This function checks to make sure the virtual address is not already mapped to a physical frame, and throws a panic if it is. Therefore, you need to write a **modified** version of this function that does not perform this check.

In summation, you should read this function and use it as a guide to map the child’s SMP virtual addresses to their corresponding physical frames.

3. Preventing SMPs from being deallocated

There is a function that deallocates a process’s page frames. You need to modify this function to prevent the SMPs from being deallocated. Only if the reference count of the dynamic SMP drops to 0 should it be deallocated.

(to be continued)

4. Dynamically mapping the dynamic SMP when a process first writes to it

When a process first tries to write to the dynamic SMP, it is read-only so a page fault **trap** will be thrown. You need to add a handler for this that:

1. Allocates a physical frame for the dynamic SMP
2. Maps all of the processes’ virtual addresses of the dynamic SMP to this physical frame

(to be continued)