PROJECT 2: USER PROGRAMS DESIGN DOCUMENT

Zhang Yichi zhangych6@shanghaitech.edu.cn
Hu Aibo huab@shanghaitech.edu.cn

PRELIMINARIES

If you have any preliminary comments on your submission, notes for the TAs, or extra credit, please give them here.

Please cite any offline or online sources you consulted while preparing your submission, other than the Pintos documentation, course text, lecture notes, and course staff.

PAGE TABLE MANAGEMENT

DATA STRUCTURES

A1: Copy here the declaration of each new or changed `struct` or `struct` member, global or static variable, `typedef`, or enumeration. Identify the purpose of each in 25 words or less.

ALGORITHMS

A2: In a few paragraphs, describe your code for accessing the data stored in the SPT about a given page.

A3: How does your code coordinate accessed and dirty bits between kernel and user virtual addresses that alias a single frame, or alternatively how do you avoid the issue?

SYNCHRONIZATION

A4: When two user processes both need a new frame at the same time, how are races avoided?

RATIONALE

A5: Why did you choose the data structure(s) that you did for representing virtual-to-physical mappings?

PAGING TO AND FROM DISK

DATA STRUCTURES

B1: Copy here the declaration of each new or changed `struct` or `struct` member, global or static variable, `typedef`, or enumeration. Identify the purpose of each in 25 words or less.

ALGORITHMS

B2: When a frame is required but none is free, some frame must be evicted. Describe your code for choosing a frame to evict.

B3: When a process P obtains a frame that was previously used by a process Q, how do you adjust the page table (and any other data structures) to reflect the frame Q no longer has?

B4: Explain your heuristic for deciding whether a page fault for an invalid virtual address should cause the stack to be extended into the page that faulted.

SYNCHRONIZATION

B5: Explain the basics of your VM synchronization design. In particular, explain how it prevents deadlock. (Refer to the textbook for an explanation of the necessary conditions for deadlock.)

B6: A page fault in process P can cause another process Q's frame to be evicted. How do you ensure that Q cannot access or modify the page during the eviction process? How do you avoid a race between P evicting Q's frame and Q faulting the page back in?

B7: Suppose a page fault in process P causes a page to be read from the file system or swap. How do you ensure that a second process Q cannot interfere by e.g. attempting to evict the frame while it is still being read in?

B8: Explain how you handle access to paged-out pages that occur during system calls. Do you use page faults to bring in pages (as in user programs), or do you have a mechanism for "locking" frames into physical memory, or do you use some other design? How do you gracefully handle attempted accesses to invalid virtual addresses?

RATIONALE

B9: A single lock for the whole VM system would make synchronization easy, but limit parallelism. On the other hand, using many locks complicates synchronization and raises the possibility for deadlock but allows for high parallelism. Explain where your design falls along this continuum and why you chose to design it this way.

MEMORY MAPPED FILES

DATA STRUCTURES

C1: Copy here the declaration of each new or changed `struct` or `struct` member, global or static variable, `typedef`, or enumeration. Identify the purpose of each in 25 words or less.

ALGORITHMS

C2: Describe how memory mapped files integrate into your virtual memory subsystem. Explain how the page fault and eviction processes differ between swap pages and other pages.

C3: Explain how you determine whether a new file mapping overlaps any existing segment.

RATIONALE

C4: Mappings created with "mmap" have similar semantics to those of data demand-paged from executables, except that "mmap" mappings are written back to their original files, not to swap. This implies that much of their implementation can be shared. Explain why your implementation either does or does not share much of the code for the two situations.