CS 130 Operating Systems 1
Homework Assignment #2
(Due at 11:59pm Dec. 13, 2020)

- Please type the solutions using WORD, LaTex, etc, or write by hand very neatly and legibly, comparable to typing
- Please pay special attention to the DUE DATE – no late turn in or special case consideration
- Please submit your homework as a PDF file to Gradescope
- Please work on your homework individually
- The file name should be in a form of "id-YourName-hw2", for example "12345-ZhangSan-hw2.pdf"

1. [15 points] **Replacement Policy**
Consider the following page reference string:

a, b, c, d, a, b, d, c, c, b, a

With 3 frames, how many page faults would occur with the following page replacement algorithms? Fill in the tables accordingly.

Hint: all frames are initially empty, so your first unique pages will all cost one fault each.

1.1. FIFO

Page	A	B	C	D	A	B	D	C	B	A
1										
2										
3										

How many misses will you get with FIFO?

Page	A	B	C	D	A	B	D	C	B	A
1	A			D			+	C		
2		B			A					+
3			C			B			+	

7misses

1.2. LRU

Page	A	B	C	D	A	B	D	C	B	A

1										
2										
3										

How many misses will you get with LRU?

Page	A	B	C	D	A	B	D	C	B	A
1	A			D			+			A
2		B			A			C		
3			C			B			+	

8 misses

1.3. MIN

Page	A	B	C	D	A	B	D	C	B	A
1										
2										
3										

How many misses will you get with MIN?

Page	A	B	C	D	A	B	D	C	B	A
1	A				+					+
2		B				+			+	
3			C	D			+	C		

5 misses

2. [10 points] **Address Translation**
Consider a machine with a physical memory of 8 GB, a page size of 8 KB, and a page table entry size of 4 bytes. How many levels of page tables would be required to map a 46-bit virtual address space if every page table fits into a single page?

Since each PTE is 4 bytes and each page contains 8KB, then a one-page page table would point to 2048 or 2^11 pages, addressing a total of 2^11 * 2^13 = 2^24 bytes.
Depth 1 = 2^24 bytes
Depth 2 = 2^35 bytes
Depth 3 = 2^46 bytes
So in total, 3 levels of page tables are required.

3. [15 points] **Demand Paging**
An up-and-coming big data startup has just hired you do help design their new memory system for a byte-addressable system. Suppose the virtual and physical memory address space is 32 bits with a 4KB page size.

Suppose you know that there will only be 4 processes running at the same time, each with a Resident Set Size (RSS) of 512MB and a working set size of 256KB. What is the minimum amount of TLB entries that your system would need to support to be able to map/cache the working set size for one process? What happens if you have more entries? What about less?

A process has a working set size of 256KB which means that the working set fits in 64 pages. This means our TLB should have 64 entries. If you have more entries, then performance will increase since the process often has changing working sets, and it should be able to store more in the TLB.

If it has less, then it can't easily translate the addresses in the working set and performance will suffer.

4. [20 points] **Inverted Page Tables**
Consider the following case:
- 64-bit virtual address space
- 4 KB page size
- 512 MB physical memory

4.1 How much space (memory) needed for a single level page table?

Hint1: How many entries are there?
1 per virtual page.
Hint2:What is the size of a page table entry?
access control bits + physical page #.

One entry per virtual page
- 2^64 addressable bytes / 2^12 bytes per page = 2^52 page table entries

Page table entry size
- 512 MB physical memory = 2^29 bytes
- 2^29 bytes of memory/2^12 bytes per page = 2^17 physical pages
- 17 bits needed for physical page number
{ Page table entry = ~4 bytes}
- 17 bit physical page number = ~3 bytes
- Access control bits = ~1 byte
Page table size = page table entry size * # total entries
{ 2^52 page table entries * 2^2 bytes = 2^54 bytes (16 petabytes)}

4.2 Linear Inverted Page Table
What is the size of the hashtable? What is the runtime of finding a particular entry?
Assume the following:
- 16 bits for process ID
- 52 bit virtual page number (same as calculated above)
- 12 bits of access information

{ add up all bits = 80 bits = 10 bytes
- 10 bytes * # of physical pages = 10 * 2^17 = 2^3 * 2^17 = 1 MB
Iterate through all entries.
For each entry in the inverted page table,
compare process ID and virtual page
number in entry to the requested process
ID and virtual page number
Extremely slow. must iterate through 2^17 entries of the hash table worst-case scenario.

5. [40 points] **File System**
5.1 (1)Calculate the maximum file size for a file in FAT.
(2)Calculate the maximum file size for a file in the Unix file system(FFS).
(Assume a block size of 4KiB. In FAT, assume file sizes are encoded as 4 bytes. In FFS block pointers are 4 bytes long.)

FAT : 4GB
FFS : Since a disk block is 4KB (212 bytes) and a block number is 4

(2^2) bytes, there are 2^10 = 1024 entries per indirect block. Therefore, the maximum number of blocks of a file that could be referenced by the i-node is 12 + 2^10 + (2^10)^2 + (2^10)^3 = 12 + 2^10 + 2^20 + 2^30. Thus, the maximum size of the file would be (12+2^10 +2^20 +2^30)×2^12 = (12×2^12)+2^22 +2^32 +2^42 = 48KB+4MB+4GB+4TB.

5.2 (1) What are the advantages of an inode-based file system design compared to FAT?

Fast random access to files. Support for hard links.

(2) Why do we have direct blocks? Why not just have indirect blocks?

Faster for small files.

(3) Consider a file system with 2048 byte blocks and 32-bit disk and file block pointers. Each file has 12 direct pointers, a singly-indirect pointer, a doubly-indirect pointer, and a triply-indirect pointer. How large of a disk can this file system support?

2^32 blocks x 2^11 bytes/block = 2^43 = 8 Terabytes.

5.3 Rather than writing updated files to disk immediately, many UNIX systems use a delayed write- behind policy in which dirty disk blocks are flushed to disk once every x seconds. List two advantages and one disadvantage of such a scheme.

Advantage 1: The disk scheduling algorithm (i.e. SCAN) has more dirty blocks to work with at any one time and can thus do a better job of scheduling the disk arm.
Advantage 2: Temporary files may be written and deleted before data is written to disk. Disadvantage: File data may be lost if the computer crashes before data is written to disk.