Project 4: File Systems

Preliminaries

> Fill in your name and email address.

FirstName LastName email@domain.example

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

Well, I have believed that system research is really a tough work

And concurrency is more intractable than the OS architecture design.

So when we will have rust OS in PKU course? Have you started working on it?

> 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.

Indexed And Extensible Files

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.

All the remaining unused data are used to save four-byte long indirect sector numbers. The unified visiting mode is easy to implement.

If we need to find target sector of offset `position` in a file, we firstly find the first indirect index `sub`. And then the target sector is set on the sector with id `sub`.

> A2: What is the maximum size of a file supported by your inode structure? Show your work.

There are `123` indirect blocks. Each block saves (512 / 4) = `128` direct blocks. So the maximum size of a file is 123 * 128 * 512 bytes = 7.67875MB

SYNCHRONIZATION

> A3: Explain how your code avoids a race if two processes attempt to extend a file at the same time.

Every file can only be written by only one process at a time. Protected by a lock.

> A4: Suppose processes A and B both have file F open, both
> positioned at end-of-file. If A reads and B writes F at the same
> time, A may read all, part, or none of what B writes. However, A
> may not read data other than what B writes, e.g. if B writes
> nonzero data, A is not allowed to see all zeros. Explain how your
> code avoids this race.

When B extends the length of the file F, A can not read the file. (When there is a writer running, no reader can read the file.)

After B's writing, A can not read across the border of the file.

> A5: Explain how your synchronization design provides "fairness".
> File access is "fair" if readers cannot indefinitely block writers
> or vice versa. That is, many processes reading from a file cannot
> prevent forever another process from writing the file, and many
> processes writing to a file cannot prevent another process forever
> from reading the file.

Maintain a queue. Every time a reader comes to the front of the queue, if there are some reader reading the file, the new reader can join them. But if there is a writing change the content of the file, it must wait.

Every time a writer comes to the front of the queue, it must wait all the current reader finishing their jobs and do its change exclusively. (A writer also wait the writer changing the file at present)

RATIONALE

> A6: Is your inode structure a multilevel index? If so, why did you
> choose this particular combination of direct, indirect, and doubly
> indirect blocks? If not, why did you choose an alternative inode
> structure, and what advantages and disadvantages does your
> structure have, compared to a multilevel index?

I only use the indirect blocks, absolutely, for easy implementation. It can be upgraded easily and is not a hard part in the file system. If you want to give me a F because of this, I can change it as soon as possible. But anyway, the Pintos has a highly-coupling architecture as a macro kernel. As you can see, the kernel is divided into thread, userprog, vm and filesys just for teaching. It is not a good idea to design a real-world os like this. And we need some object-oriented design pattern to help us build a OS with higher maintainability. Concurrency is also a intractable problem. So when are you to startup the project of PKUOS in rust version? C language has a bit of out-of-date now.

Subdirectories

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.

A directory is treated like file. Some differences between the ordinary file and directory fall in the content of `inode_disk` sector. We write the type of inode `(INODE_DIR | INODE_FILE)` and the entry number in the metadate sector(inode).

The directory "file" save all the entries of the directory. 20 bytes per entry from the zero offset of the "file". The first two entry are `.` and `..`

ALGORITHMS

> B2: Describe your code for traversing a user-specified path. How
> do traversals of absolute and relative paths differ?

The link relationship of directory, subdirectory and file forms a tree. The process of traversing a path starts from one directory inode, search the next layer name in the directory entry and go to the next directory inode. But common files and directories are different in type and files do not have directory entry structure. So except for the last layer, we should check the file type of inode, the type must be `INODE_DIR`.

The absolute and relative path can be traversed in the same logic thanks to the `.` and `..` entries.

SYNCHRONIZATION

> B3: How do you prevent races on directory entries? For example,
> only one of two simultaneous attempts to remove a single file
> should succeed, as should only one of two simultaneous attempts to
> create a file with the same name, and so on.

The remove of file is done by changing the remove boolean variable in inode(in-memory). After the `open_cnt` turning to zero, the file will be removed in the last reference closing process.

The create of file is also a writing process to the directory file. The file writing is exclusive. So the second creation will fail.

> B4: Does your implementation allow a directory to be removed if it
> is open by a process or if it is in use as a process's current
> working directory? If so, what happens to that process's future
> file system operations? If not, how do you prevent it?

This operation is allowed. In the future operations, the user process can not reopen the removed directory. This is prevented by `inode_reopen` returning `NULL` pointer if the remove mark in inode is true.

RATIONALE

> B5: Explain why you chose to represent the current directory of a
> process the way you did.

Save a `struct dir` pointer in the struct of thread. If the current directory changes, the old directory will be closed.

Buffer Cache

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.

The cache is implemented as a indirect layer on the block. All the block operations in the `inode.c` are replaced by cache operations. It corresponds the philosophy of computer system design: *All problems in computer science can be solved by another level of indirection*

ALGORITHMS

> C2: Describe how your cache replacement algorithm chooses a cache
> block to evict.

Second chance. If the cached block has been accessed recently, the `CACHE_A` bit will be set to 1. And we set `CACHE_A` to zero if we want to find a cached block to evict. The block misses its "second chance" will be drove away poorly.

> C3: Describe your implementation of write-behind.

The dirty block is kept in the cache. If we decide to evict some cached block, we will check if it is dirty by `CACHE_D` bit in the cache metadata. If it is dirty, the block writing are executed.

> C4: Describe your implementation of read-ahead.

When allocate the block to the file, we extend the file length by some continuous blocks. So we can just read the next block. If we want to read the next block of the file, it needs another parameter and two block reading. The implementation will be inelegant.

SYNCHRONIZATION

> C5: When one process is actively reading or writing data in a
> buffer cache block, how are other processes prevented from evicting
> that block?

The `CACHE_A` bit will set to one after every visit. If we visit the block frequently, it is less likely to be evicted.

> C6: During the eviction of a block from the cache, how are other
> processes prevented from attempting to access the block?

Another bit in cache metadata flag, `CACHE_PIN` can prevent it from evicting when we try to find a victim. After the reading or writing, the `CACHE_PIN` will set to zero again.

RATIONALE

> C7: Describe a file workload likely to benefit from buffer caching,
> and workloads likely to benefit from read-ahead and write-behind.

Assume that a file is created and a file is writing something to extend its length. Many other processes are reading this file. The writes and reads can be mainly done in the memory. It is faster than many operations to read blocks on the disk.

Something else

Hooray! I finish my burden of OS class... 8 days earlier. (I should finish it earlier. Sigh.)

I have a sense of being a deserter when choosing PF. But I really do not get a good score in the midterm exam relative to my classmates who want to improve GPA in OS class. Exam brings me great pain. In fact the only exam I got good marks is PKU winter camp of Olympic Information, so I am here. How I wish that I did not engage in the information competition and went to an ordinary 985 by my score in college entrance exam. Sometime I think all the things I have got are given in charity because girls are scarce in CS. They let me enter the PKU and be a student in Turing Class. I hate this destiny. A person knows my grade in Turing admission exam asked a professor that is my qualification to Turing because I am female. Well, I always have a deepest idea to prove that I can do something really excellent to fit in with WHAT I HAVE. I learned OS for a long time in the winter holiday, well, finally I become a deserter.

Oh, sorry, I talk too much about myself. Actually I want to tell you something, my TA.

I find you are anxious about your doctorate degree. Some peer pressure is strong. But I thought you are different from your group members. Students in Peking University generally do not have the consciousness of cooperation. But the era of individual heroism has passed, a useful project needs effort of a large group of people. You are more earnest than members in your research group, but you still lack the experience of cooperation. Although the unknown difficulty of achieving degree hinders your way to devote yourself to computer science education, I know your talent in computer system is to help you design a effective and secure system in the near future. It comes to me that the most important part in computer system is to find a problem. I hope you have found a meaningful problem now. With your prominent ability, I believe that it is easy to get the degree within two years.

So, what I really want to say is: if you want to start your big project and need a group, it is my pleasure to be there for you.

PS: Prof Jin may need a syllabus with a clear roadmap. The way of introducing a new topic is confusing. The syllabus of textbook in Shanghai Jiaotong University may be a good assistant.

PSS: Thank you for your excellent job as a TA. You are really patient with my weird problems. I have realized that I have too much to learn. Hope you become an excellent researcher in system and enjoy your fame over the world one day.

PSSS: I don't write the words above with intention of getting P. But I will be happy if you are willing to assign my score a P.

PSSSS: If you decide to give me a F before 6.14 8:30, please tell me as soon as possible. I will take the final exam.

PSSSSS: If you give me a F quietly and I have to see Xin Jin again next year, well, that's OK. Because Xin Jin is a handsome young teacher indeed.

PSSSSSS: But Anyway, Qun Huang is more handsome. Actually the most handsome man in my heart. No one can beat him.