BÀI TOÁN: XÂY DỰNG APIs XỬ LÝ FILE TEXT VỚI CÁC CHỨC NĂNG NHƯ CHÈN, SỬA, XÓA DÒNG BẤT KỲ TRONG FILE, …
YÊU CẦU: TỐI ƯU BỘ NHỚ VÀ THỜI GIAN XỬ LÝ

1. int filehandle_insert(char* filename, int line_num, const char *buffer)

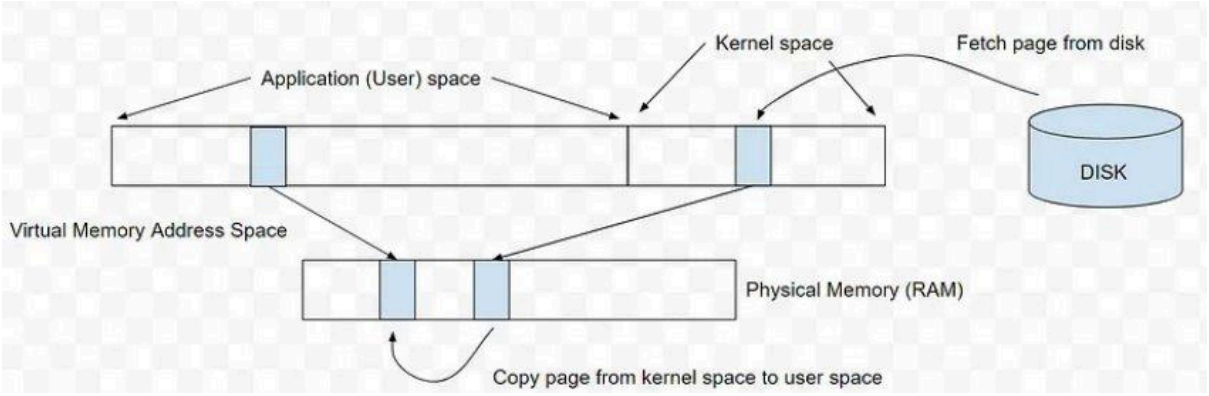
- API có chức năng chèn chuỗi ký tự từ buffer vào dòng bất kỳ trong file.
- Ý tưởng thực hiện:
    1. Duyệt các ký tự từ đầu file đến đầu dòng cần chèn.
    2. Tính toán số lượng ký tự cần dịch xuống 1 dòng, tính từ vị trí cần chèn đến cuối file và vị trí dịch đến.
    3. Sử dụng một mảng tạm có kích thước cố định (giả sử 1024 bytes) để lưu trữ lần lượt các dữ liệu cần dịch chuyển
    4. Thực hiện vòng lặp dịch chuyển lần lượt các dữ liệu đến vị trí đã tính trước, bắt đầu dịch các dữ liệu ở cuối file.
    5. Chèn chuỗi ký tự từ buffer vào vị trí cần chèn.
- Ưu điểm: Có thể kiểm soát lượng RAM sử dụng bằng cách điều chỉnh kích thước của mảng tạm.
- Nhược điểm:
    + Tốn thời gian xử lý khi phải duyệt lần lượt các ký tự từ đầu file để tìm vị trí chèn.
    + Mảng tạm có kích thước càng nhỏ thì số vòng lặp để dịch chuyển dữ liệu càng lớn.

2. int filehandle_insert_v2(char* filename, int line_num, const char *buffer)

- Ý tưởng thực hiện:
    1. Duyệt các ký tự từ đầu file đến đầu dòng cần chèn.
    2. Tính toán số lượng ký tự cần dịch xuống 1 dòng, tính từ vị trí cần chèn đến cuối file và vị trí dịch đến.
    3. Malloc 1 buffer tạm có kích thước bằng với kích thước dữ liệu cần dịch chuyển.
    4. Dịch chuyển dữ liệu
    5. Chèn chuỗi ký tự từ buffer vào vị trí cần chèn.
- Ưu điểm: Chỉ cần dịch dữ liệu 1 lần mà không cần sử dụng tới vòng lặp.
- Nhược điểm:
    + Tốn thời gian xử lý khi phải duyệt lần lượt các ký tự từ đầu file để tìm vị trí chèn.
    + Lượng RAM sử dụng không cố định, phụ thuộc vào lượng dữ liệu cần dịch chuyển.

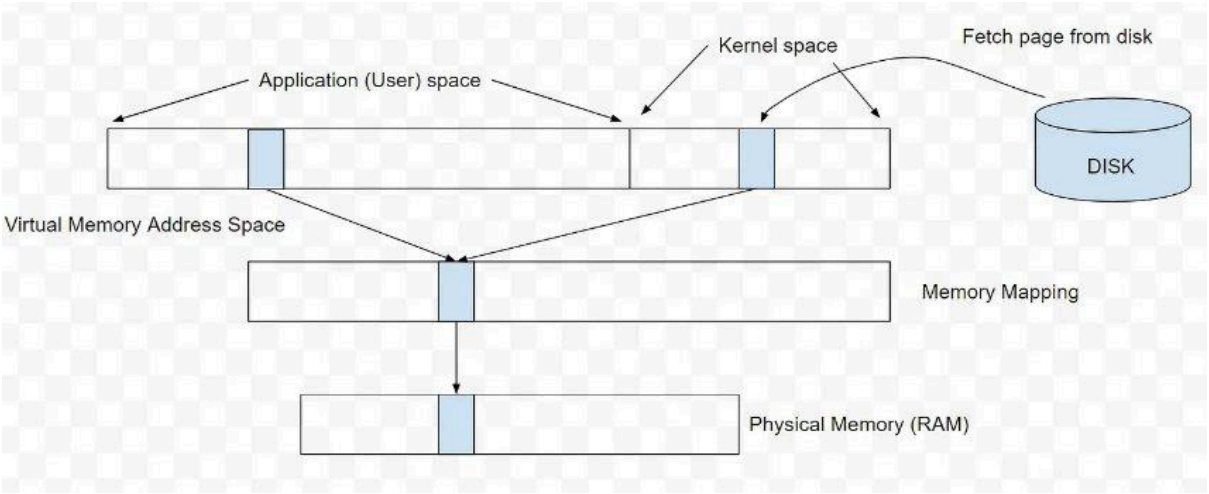
3. int filehandle_mmap_insert(char* filename, int line_num, const char *buffer)

- Sử dụng phương pháp Memory Mapped File
  2 APIs trước sử dụng các thao tác I/O tiêu chuẩn như fwrite, fread, …



Ví dụ khi gọi lệnh fread(), OS sẽ tìm và các bytes từ file trong đĩa và lưu trữ dữ liệu vào bộ đệm của kernel space. Sau đó sẽ tạo 1 bản sao để lưu vào user space. Quy trình lặp đi lặp lại khi gọi các lệnh như read, write, … dẫn đến tốn thời gian xử lý

Memory mapped file ánh xạ nội dung file vào bộ nhớ của tiến trình (cũng có thể là bộ nhớ ảo). Cho phép chương trình thao tác với dữ liệu của file một cách trực tiếp. Điều này khiến việc truy cập dữ liệu nhanh hơn, giảm thiểu số lần gọi hệ thống, tránh việc gây tốn thời gian xử lý.



- Ý tưởng thực hiện:
    1. Ánh xạ toàn bộ nội dung file vào RAM, dữ liệu được sắp xếp thành 1 mảng liên tục.
       Ví dụ nội dung file có dạng như sau:
       Viettel
       High
       Tech.
       Sau khi ánh xạ ta được 1 mảng: Viettel\nHigh\nTech.
    2. Duyệt mảng để tìm vị trí cần chèn

3. mmap có hàm để dịch chuyển các kí tự phía sau ra một đoạn vừa đủ để chèn dữ liệu mới vào
4. Chèn dữ liệu từ buffer vào mảng.

- Ưu điểm:
- Nhược điểm:
    + Tốn thời gian xử lý khi phải duyệt lần lượt các ký tự từ đầu file để tìm vị trí chèn.
    + Kích thước RAM sử dụng phụ thuộc vào kích thước file.

SO SÁNH CÁC PHƯƠNG PHÁP

- Test chức năng chèn chuỗi vào các vị trí đầu, giữa, cuối của file.
- Sử dụng pmap để kiểm tra lượng RAM mà tiến trình sử dụng và đặt log đo thời gian các APIs thực hiện.

Kết quả test với file có kích thước khoảng 100 KB

Test	Vị trí chèn	APIs		
		filehandle_insert	filehandle_insert_v2	filehandle_mmap_insert
Time (ms)	Đầu file	0.55 - 1.8	0.9 - 1.1	~ 0.15
	Giữa file	1.5 - 2	1 - 1.3	~ 0.3
	Cuối file	1.4 - 2	0.6 - 1.6	~ 0.6
Mem (KB)	Đầu file	1024	112	104
	Giữa file	1024	76	104
	Cuối file	1024	12	104

Kết quả test với file có kích thước khoảng 6.6 MB

Test	Vị trí chèn	APIs		
		filehandle_insert	filehandle_insert_v2	filehandle_mmap_insert

Time (ms)	Đầu file	30 - 40	20 - 33	4 - 8
	Giữa file	32 - 45	30 - 40	13 - 18
	Cuối file	32 - 46	37 - 55	15 - 20
Mem (KB)	Đầu file	1024	6424	6412
	Giữa file	1024	3120	6412
	Cuối file	1024	1280	6412

Các APIs đã xây dựng trong module

API	Test cases	Chức năng
int filehandle_get_total_lines(const char *filename)	7	Lấy tổng số dòng của file
int filehandle_get_line_content(const char *filename, int line_num, char *line_content)	6	Lấy ra nội dung của dòng
int filehandle_insert(char* filename, int line_num, const char *buffer)	9	Chèn chuỗi vào 1 dòng trong file
int filehandle_insert_v2(char* filename, int line_num, const char *buffer)	9	
int filehandle_mmap_insert(char* filename, int line_num, const char *buffer)	9	
int filehandle_remove(char* filename, int line_num)	7	Xóa 1 dòng bất kỳ trong file
int filehandle_mmap_remove(char* filename, int line_num)	7	
int filehandle_fix(char* filename, int line_num)	8	Sửa nội dung 1 dòng trong file
int filehandle_mmap_fix(char* filename, int line_num)	8	