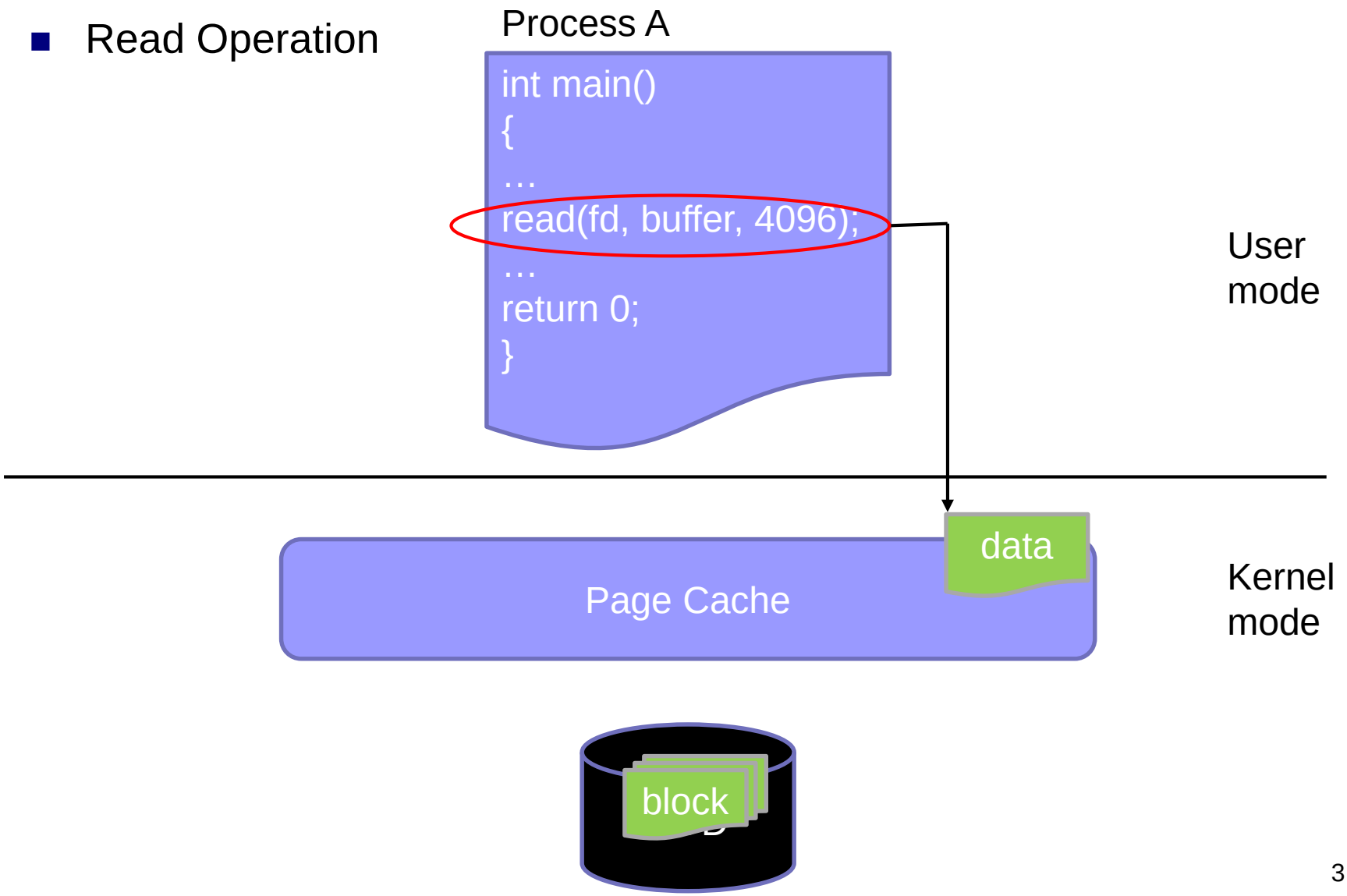
# Homework Assignment #1: I/O Performance under Sequential and Random R/W Workloads

# Outline

- **File Input / Output**
- Application Programming Interface
  - POSIX File I/O
  - Other APIs
- Homework Assignment #1
- Reference

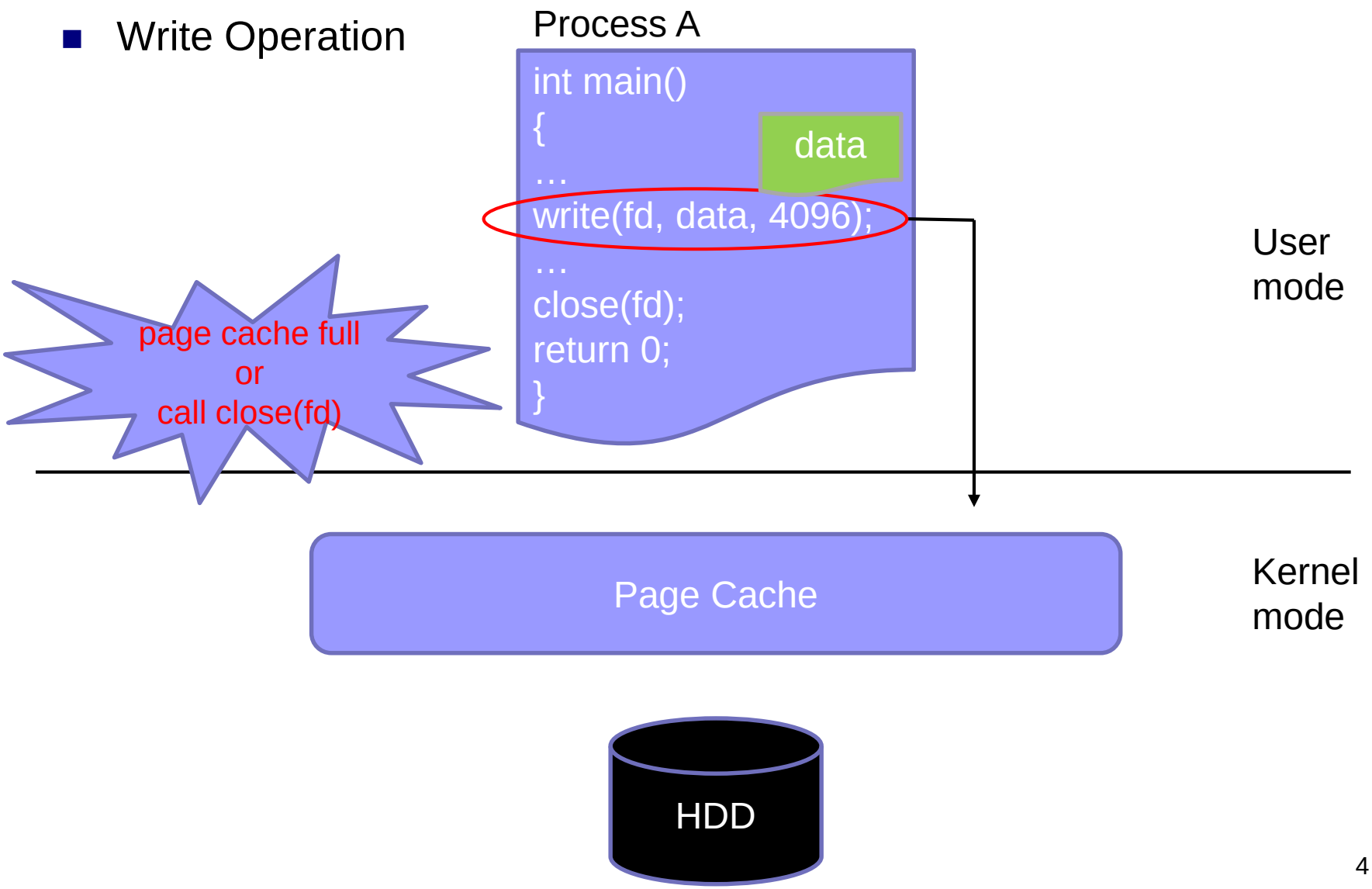
# File Input / Output

- Read Operation



# File Input / Output

- Write Operation



# Outline

- File Input / Output
- **Application Programming Interface**
  - POSIX File I/O
  - Other APIs
- Homework Assignment #1
- Reference

# POSIX File I/O

- We will use the following POSIX File I/O system calls
  - int open(const char *pathname, int flags, mode_t mode)
  - ssize_t read(int fd, void *buf, size_t count)
  - ssize_t write(int fd, const void *buf, size_t count)
  - off_t lseek(int fd, off_t offset, int whence)
  - int close(int fd);

# POSIX File I/O (cont.)

- **Open File**

```
#include <sys/types.h>
#include <sys/stat.h>
#include <fcntl.h>
```

```
int open(const char *pathname, int flags, ... /* mode_t mode */);
```

Returns file descriptor on success, or −1 on error

ex. : int fd = open( "*~/hw1/filename*", *O_RDONLY*, *S_IRWXU* );

***pathname*** : file path.

***flags*** : The flags argument is a bit mask that specifies the access mode for the file

***mode*** : the mode bit-mask argument specifies the permissions to be placed on the file

# POSIX File I/O (cont.)

- **Open File (cont.)**

```
int open(const char *pathname, int flags, ... /* mode_t mode */);
```

Returns file descriptor on success, or −1 on error

| Flag | Purpose | SUS? |
|---|---|---|
| O_RDONLY | Open for reading only | v3 |
| O_WRONLY | Open for writing only | v3 |
| O_RDWR | Open for reading and writing | v3 |
| O_CLOEXEC | Set the close-on-exec flag (since Linux 2.6.23) | v4 |
| O_CREAT | Create file if it doesn't already exist | v3 |
| O_DIRECT | File I/O bypasses buffer cache | |
| O_DIRECTORY | Fail if *pathname* is not a directory | v4 |
| O_EXCL | With O_CREAT: create file exclusively | v3 |
| O_LARGEFILE | Used on 32-bit systems to open large files | |
| O_NOATIME | Don't update file last access time on *read()* (since Linux 2.6.8) | |
| O_NOCTTY | Don't let *pathname* become the controlling terminal | v3 |
| O_NOFOLLOW | Don't dereference symbolic links | v4 |
| O_TRUNC | Truncate existing file to zero length | v3 |
| O_APPEND | Writes are always appended to end of file | v3 |
| O_ASYNC | Generate a signal when I/O is possible | |
| O_DSYNC | Provide synchronized I/O data integrity (since Linux 2.6.33) | v3 |
| O_NONBLOCK | Open in nonblocking mode | v3 |
| O_SYNC | Make file writes synchronous | v3 |

# POSIX File I/O (cont.)

- **Read File**

```
#include <unistd.h>
```

```
ssize_t read(int fd, void *buffer, size_t count);
                        Returns number of bytes read, 0 on EOF, or −1 on error
```

ex. : int ret = read( *fd*, *buffer*, *4096*);

***fd*** : file descriptor.

***buffer*** : The buffer argument supplies the address of the memory buffer into which the input data is to be placed.

***count*** : The count argument specifies the maximum number of bytes to read.

# POSIX File I/O (cont.)

- **Write File**

```
#include <unistd.h>
```

```
ssize_t write(int fd, void *buffer, size_t count);
                                Returns number of bytes written, or −1 on error
```

ex. : int ret = write( *fd*, *newData*, *4096*);

***fd*** : file descriptor.

***buffer*** : buffer is the address of the data to be written.

***count*** : count is the number of bytes to write from buffer

# POSIX File I/O (cont.)

- **Changing the File Offset**

```
#include <unistd.h>
```

```
off_t lseek(int fd, off_t offset, int whence);
                                Returns new file offset if successful, or −1 on error
```

ex. : int curr = lseek( *fd*, *0*, *SEEK_CUR*);

***fd*** : file descriptor.

***offset*** : The offset argument specifies a value in bytes.

***whence*** : The whence argument indicates the base point from which offset is to be interpreted

# POSIX File I/O (cont.)

- **Changing the File Offset (cont.)**

```
off_t lseek(int fd, off_t offset, int whence);
```

Returns new file offset if successful, or −1 on error

`SEEK_SET`

The file offset is set *offset* bytes from the beginning of the file.

`SEEK_CUR`

The file offset is adjusted by *offset* bytes relative to the current file offset.

`SEEK_END`

The file offset is set to the size of the file plus *offset*. In other words, *offset* is interpreted with respect to the next byte after the last byte of the file.

# POSIX File I/O (cont.)

- **Close File**

```
#include <unistd.h>
```

```
int close(int fd);

                                    Returns 0 on success, or −1 on error
```

ex. : int ret = close( *fd* );

***fd*** : file descriptor.

# Other APIs

- **Allocate Aligned Memory**

```
#include <stdlib.h>
```

```
void *valloc(size_t size)

    return a pointer to the allocated memory, or NULL if the request fails
```

```
ex. : char *buffer = valloc( sizeof(char) * 4096 );
```

***size*** : allocates size bytes.

The memory address will be a multiple of the page size.

# Other APIs (cont.)

- **Messure Time**

```
#include <sys/time.h>
```

```
int gettimeofday( struct timeval *tv, struct timezone *tz );
```

```
struct timeval {
        time_t tv_sec;          /* seconds */
        suseconds_t tv_usec;    /* microseconds */
};

struct timezone {
        int tz_minuteswest;     /* minutes west of Greenwich */
        int tz_dsttime;         /* type of DST correction */
};
```

# Other APIs (cont.)

- Messure Time (cont.)

```
int main()
{
        struct  timeval start;
        struct  timeval end;

        unsigned  long diff;
        gettimeofday(&start,NULL);
        delay(10);
        gettimeofday(&end,NULL);
        diff = 1000000 * (end.tv_sec - start.tv_sec) +
                                end.tv_usec - start.tv_usec;
        printf("the difference is %ld (us)\n", diff);
        return 0;
}
```

# Other APIs (cont.)

- **rand()**
  - rand() function is provided by stdlib.h library.
  - Returns a pseudo-random number in the range of 0 to RAND_MAX.

```
#include<stdlib.h>
```

```
int rand(void);
```

```
ex. : rand();
```

# Other APIs (cont.)

- **srand()**
  - srand() function is provided by stdlib.h library.
  - This function seeds the random number generator used by the function rand().

```
#include<stdlib.h>
```

```
void srand(unsigned int seed);
```

```
ex. : srand(1000);
```

# Other APIs (cont.)

- **time()**
  - time() function is provided by time.h library.
  - Calculates the current calender time and encodes it into time_t format.

```
#include<time.h>
```

```
time_t time(time_t *t);
```

```
ex. : time(NULL);
```

# Example

```
 5 #include <stdio.h>
 6 #include <stdlib.h>
 7 #include <sys/types.h>  //open()
 8 #include <sys/stat.h>   //open()
 9 #include <fcntl.h>      //open()
10 #include <unistd.h>     //read()
11 #include <sys/time.h>   //gettimeofday()
12 #include <stdint.h>
13
14 int main()
15 {
16         int32_t fd;
17         int8_t *buffer = valloc(sizeof(char) * 4096);
18         int32_t ret;
19         int64_t exTime;
20         struct timeval timeStart;
21         struct timeval timeEnd;
22
23         fd = open("data", O_RDONLY);
24         if (fd == -1){
25                 printf("open error\n");
26                 return 0;
27         }
28         gettimeofday(&timeStart, NULL);
29         ret = read(fd, buffer, 4096);
30         gettimeofday(&timeEnd, NULL);
31         if (ret == -1){
32                 printf("file read error\n");
33                 return 0;
34         }
35         free(buffer);
36         close(fd);
37
38         exTime = 1000000 * (timeEnd.tv_sec - timeStart.tv_sec) + (timeEnd.tv_usec - timeStart.tv_usec);
39
40         printf("exTime = %ld us\n", exTime);
41         return 0;
42 }
```

# Random number example

```
#include <stdio.h>
#include <stdlib.h>
#include <time.h>
void main()
{
 int a;
 srand(time(NULL));
 for(int i=0; i<=5; i++){
            a=(rand()%100)+1;
            printf("The Random Number is %d \n", a);
 }
}
```

```
arthur@arthur-VirtualBox:~/hw2$ ./rand
The Random Number is 14
The Random Number is 87
The Random Number is 7
The Random Number is 27
The Random Number is 98
The Random Number is 37
```

# Outline

- File Input / Output
- Application Programming Interface
  - POSIX File I/O
  - Other APIs
- **Homework Assignment #1**
- Reference

# Homework Assignments #1: Sequential

- 1st: Write a program that creates a 100MB file on your local disk
- 2nd: Measure the time of following three I/O access scenarios
- Sequential Read / Write (30%)
  - Sequential Read:
    - Read the file sequentially by reading the file from beginning to end.
  - Sequential Write:
    - Overwrite the file with 100MB of new data by writing the file from beginning to end.
  - Sequential Write (using O_DIRECT):
    - Overwrite the file with 100MB of new data by writing the file using O_DIRECT flag from beginning to end.

# Homework Assignments #1: Random

- 1st: Write a program that creates a 100MB file on your local disk
- 2nd: Measure the time of following three I/O access scenarios
- Random Read / Write(Do 50,000 times) (30%)
  - Random Read:
    - Choose 4KB-aligned offset in the file uniformly at random, seek to that location in the file, and read 4KB of data at that position.
  - Random Write:
    - Choose 4KB-aligned offset in the file uniformly at random, seek to that location in the file, and overwrite the file with 4KB of new data at that position.
  - Random Write (using O_DIRECT):
    - Choose 4KB-aligned offset in the file uniformly at random, seek to that location in the file, and overwrite the file using O_DIRECT flag with 4KB of new data at that position.

# Homework Assignments #1

- Sequential Read / Write (30%)
  - each program 10%
- Random Read / Write (30%)
  - each program 10%
- Word (40%)
  - Explain your execution times.
    1. Compare and explain your results of Sequential Read, Sequential Write and (O_DIRECT) Sequential Write. (8%)
    2. Compare and explain your results of Random Read, Random Write and (O_DIRECT) Random Write. (8%)
    3. Compare and explain your results of Sequential Read and Random Read. (8%)
    4. Compare and explain your results of Sequential Write and Random Write. (8%)
    5. Compare and explain your results of (O_DIRECT) Sequential Write and (O_DIRECT) Random Write. (8%)

# Appendix

- Get readahead size

```
arthur@arthur-VirtualBox:~$ sudo /sbin/blockdev --getra /dev/sda
256
```

- Set readahead to 0

```
arthur@arthur-VirtualBox:~$ sudo /sbin/blockdev --setra 0 /dev/sda
```

- Clear page cache, dentries, inode
  - 1st: Sync dirty data to disk

```
arthur@arthur-VirtualBox:~$ sudo sync
```

  - 2nd: clear page cache

```
arthur@arthur-VirtualBox:~$ sudo sh -c 'echo 3 > /proc/sys/vm/drop_caches'
```

# Appendix

- How to use *O_DIRECT* flag

```
fd = open("data", O_WRONLY | O_DIRECT);
```

- Include headler

```
#include <sys/types.h>  //open()
#include <sys/stat.h>   //open()
#define __USE_GNU 1
#include <fcntl.h>      //open()
```

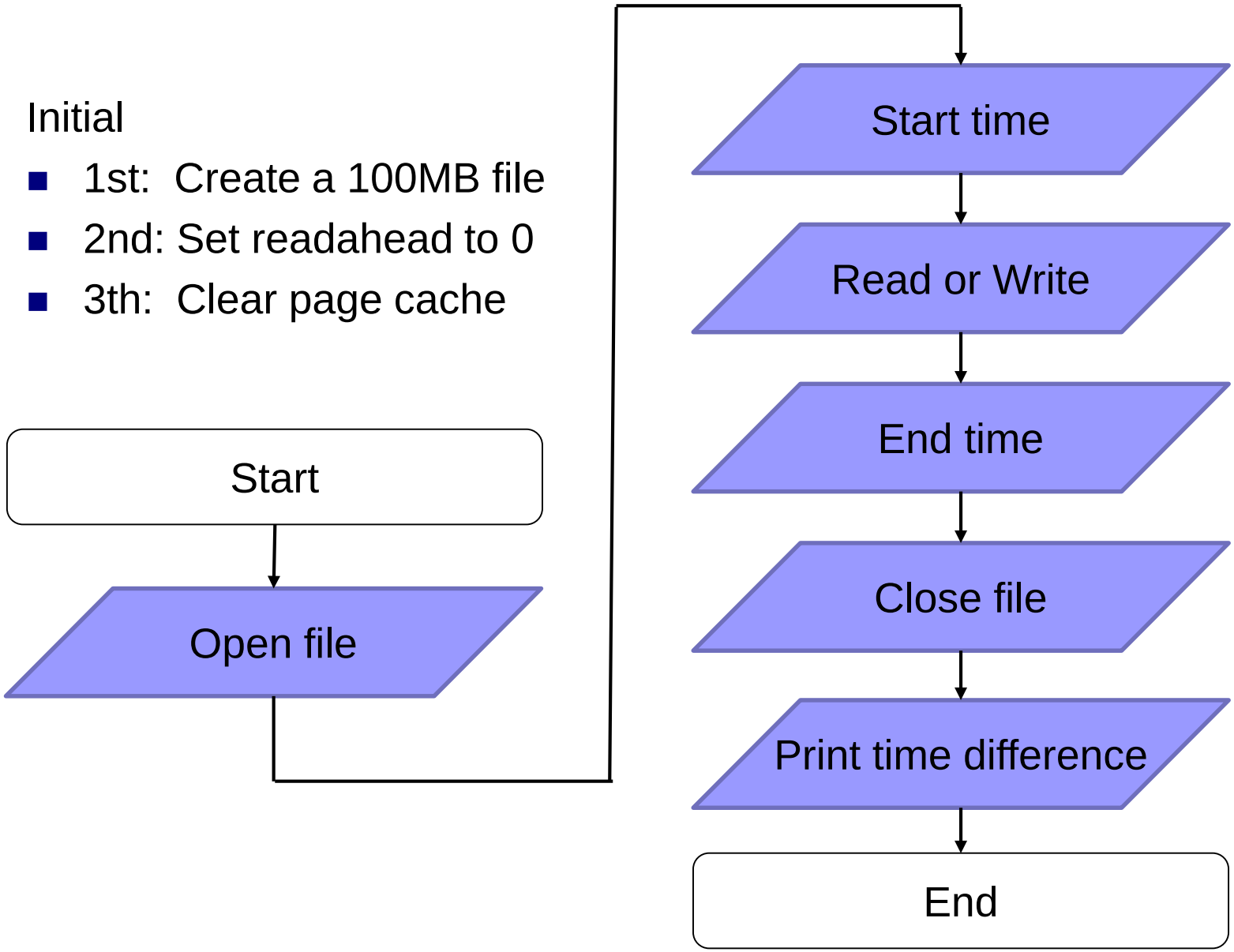
- Add *#define __USE_GUN 1 over #include <fcntl.h>*

```
#define __USE_GNU 1
#include <fcntl.h>      //open()
```

# Flowchart

Initial

- 1st: Create a 100MB file
- 2nd: Set readahead to 0
- 3th: Clear page cache



# Result

- Sequential Read

```
oslab@oslab-ASUSPRO-D840MB-M840MB:~/filesystem_hw1/hw1/seqRead$ ./seqRead.out
seqRead exTime = 1395255 us
```

- Sequential Write

```
oslab@oslab-ASUSPRO-D840MB-M840MB:~/filesystem_hw1/hw1/seqWrite$ ./seqWrite.out
seqWrite exTime = 29613 us
```

- Sequential Write (O_DIRECT)

```
oslab@oslab-ASUSPRO-D840MB-M840MB:~/filesystem_hw1/hw1/seqWrite_direct$ ./seqWrite_direct.out
seqWrite_direct exTime = 505052 us
```

# Result

- Random Read

```
oslab@oslab-ASUSPRO-D840MB-M840MB:~/filesystem_hw1/hw1/ranRead$ ./ranRead.out
ranRead exTime = 87206953 us
```

- Random Write

```
oslab@oslab-ASUSPRO-D840MB-M840MB:~/filesystem_hw1/hw1/ranWrite$ ./ranWrite.out
ranWrite exTime = 60447 us
```

- Random Write (O_DIRECT)

```
oslab@oslab-ASUSPRO-D840MB-M840MB:~/filesystem_hw1/hw1/ranWrite_direct$ ./ranWrite_direct.out
ranWrite_direct exTime = 693525402 us
```

# Turn in

- Deadline

  2020/04/08 PM.11:59:59
- Upload to ilearning
- File name
  - HW1_ID.zip (e.g. HW1_7105056035.zip)
    - Source code
      - .c file
    - Word

- If you don’t hand in your homework on time, your score will be deducted 10 points every day.

# Rules

- **No cheat work** is acceptable
  - You get zero if you copy other people's version.
- **Only single job** is accepted

# TA

- Name：Cheng-Chia, Chang
- Email：s9001055@gmail.com
  - Title format : FS HW#1 - [your name]
- Lab: OSNET(1001A)

# Reference

- Operating System Concepts, 10th Edition
- The Linux Programming Interface:
  A Linux and UNIX System Programming Handbook
- https://systemprogrammingatntu.github.io/