Name: Shijian ZHENG UID: 804514110

Lab 2 Design Problems – Change Notification

1. Description of Design Problem:

1) Brief Description:

A process P1 can open a ramdisk and request notification on any change. When another process writes to the ramdisk, P1 should receive notification. A waiting process P1 should be able to block on something, such as a system call like ioctl, until the notification arrives. The waiting process should be able to wait for changes to specific areas of the disk. For instance, maybe there will be separate notifications for each sector of the disk, or for ranges of sectors, all the way down to individual bytes. Use time and space overheads.

2) Specification:

a) A process P1 can open a ramdisk to read or write and request notification on any change. For instance, "echo a | ./osprdaccess –w –l –n" and "./osprdaccess –r –l -n" where "-n" represents that this process request the notification before it gets a lock.

b) When a process P1 request notification, it will be blocked until it receive the notification. For instance, the execution order of command, "./osprdaccess –r –l –d 5 -n", is as followed. Firstly, process P1 is blocked until the notification arrives. Secondly, the process P1 is woken up after being notified. Thirdly, P1 acquires the reading lock. Lastly, P1 reads the content on ramdisk after 5 seconds.

c) The process P1 will only be woken up when some other processes write the ramdisk. In other words, if some processes read from the ramdisk, P1 will not receive any notification.

d) Process P1 can request a notification when specific area of ramdisk is modified, not just only when the whole ramdisk is modified. For instance, maybe there will be separate notifications for each sector of the disk, or for ranges of sectors, all the way down to individual bytes.

e) Consider the situation that the process is killed while it is waiting for the notification.

2. Implementation Strategy:

1) Enable the user to use "-n" in command.

a) Detect the notification option in command.

b) When a process requests a notification, kernel will call the "ioctl" function with parameter of "OSPRDIOCNOTIFICATION".

c) When the kernel executes "osprd_ioctl" function with "OSPRDIOCNOTIFICATION", it will send a message to terminal. Therefore, we can know user has successfully used "-n" in command.

2) Process is notified when whole ramdisk is modified.

a) Add a new variable "blockq_N" to be the wait queue for tasks blocked until it is notified. Add a new variable "num_of_wait" to be the total number of the tasks

waiting for notification. Add a new variable "notification_arrived" to be the flag used to tell the process whether the notification has arrived or not.

b) Initialize the new variables defined in "struct osprd_info".

c) When a process request notification, kernel will block this process until some other processes write on the ramdisk. The notification will wakes up all the processes in the waiting queue blockq_N. After all processes were woken up, the flag "notification_arrived" is reset to 0 and the processes will read or write as the command requests.

d) Whenever there is a process finishing its writing task, it will wake up all the process waiting for the notification.

3) Process is notified when specific area of ramdisk is modified. (Part 1)

a) Allow the user to wait for the notification for whole ramdisk, particular sector, sectors' range, particular byte or bytes' range.

i) **** -n: Notification for whole ramdisk;

ii) **** -n –s x: Notification for particular sector x;

iii) **** -n –S x y: Notification for sectors' range from x to y;

iv) **** -n –b x: Notification for particular byte;

v) **** -n –B x y: Notification for bytes' range;

b) These commands can be synthesized into one command, the command requesting notification for byte's range; (SS = SECTORSIZE, NS = nsector = # of total sectors)

i) **** -n = **** -n –B 0 (NS*SS – 1);

ii) **** -n –s x = **** -n –B (x*SS) ((x + 1)*SS – 1);

iii) **** -n –S x y = **** -n –B (x*SS) ((y+1)*SS – 1);

iv) **** -n –b x = **** -n –B x x;

v) **** -n –B x y = **** -n –B x y;

c) Check if the variables can be passed successfully from the user space to kernel space.

d) In kernel program, the "offset" and "size" information of write command are needed to figure out which part of ramdisk is going to be modified.

4) Process is notified when specific area of ramdisk is modified. (Part 2)

a) When "cmd == OSPRDIOCNOTIFICATION" is true, which means that the process is requesting a notification for specific area on ramdisk, all informations like pid and that specific area are stored into the linked list "wait_notif_pids". Then, put this process into "blockq_N" and block it until other process calls "wake_up_all (blockq_N)"

b) Establish one more command for "ioctl" named "OSPRDIOCWAKENOTIFICATION", which is used to wake up the processes waiting for notification. There are two variables needed to pass into kernel space. One is the offset of writing. The other is the size of writing.

c) Whenever a writing task is done, wake up the processes whose specific area was just modified.

5) Solve the ordering issue.
   a) When multiple processes waiting for the notification are woken up at the same time, they need to their commands in order. Therefore, "ticket" variable used in lab2 can also be used here to solve the problem of ordering.
   b) Use ticket_head_N to assign the tickets to the processes which wait for notification. And use ticket_tail_N to determine which process need to be woken up firstly.

6) Solve the issue that process is killed while waiting for notification.
   a) When the process is waiting for notification, it can be killed. At this time, the "ticket" variable need to figure out whose ticket becomes invalid due to its death.
   b) Create a new linked list named "invalid_ticket_N" which use to store the processes which were killed while they are waiting for the notification.

3. Implementation Result:

1) Enable the user to use "-n" in command.
   a) "echo a | ./osprdaccess –w –l –d 1 -n";

```
debian:/tmp/cs111/lab2-shijian# echo a | ./osprdaccess -w -l -d 1 -n
Process[3173] is waiting the notification!
```

   b) "./osprdaccess –r –L –d 1 -n";

```
debian:/tmp/cs111/lab2-shijian# ./osprdaccess -r -L -d 1 -n
Process[3179] is waiting the notification!
```

2) Process is notified when whole ramdisk is modified;
   a) "echo aaa | ./osprdaccess –w –l"
   "./osprdaccess –r –l –n &"
   "./osprdaccess –r –l –n &"
   "echo bbb | ./osprdaccess –w –l";

```
debian:/tmp/cs111/lab2-shijian# echo aaa | ./osprdaccess -w -l
debian:/tmp/cs111/lab2-shijian# ./osprdaccess -r -l -n &
[1] 3171
debian:/tmp/cs111/lab2-shijian# Process[3171] is waiting the notification!
./osprdaccess -r -l -n &
[2] 3172
debian:/tmp/cs111/lab2-shijian# Process[3172] is waiting the notification!
echo bbb | ./osprdaccess -w -l
Process[3172] has received the notification!
Process[3171] has received the notification!
bbb
bbb
```

   b) "echo aaa | ./osprdaccess –w –l"
   "./osprdaccess –r –l –n &"
   "./osprdaccess –r –l &"
   "echo bbb | ./osprdaccess –w –l";

```
debian:/tmp/cs111/lab2-shijian# echo aaa | ./osprdaccess -w -l
debian:/tmp/cs111/lab2-shijian# ./osprdaccess -r -l -n &
[1] 3181
debian:/tmp/cs111/lab2-shijian# Process[3181] is waiting the notification!
./osprdaccess -r -l &
[2] 3182
debian:/tmp/cs111/lab2-shijian# aaa
echo bbb | ./osprdaccess -w -l
Process[3181] has received the notification!
bbb
```

3) Process is notified when specific area of ramdisk is modified. (Part 1)
   a) Test case for checking the variables "Bx" and "By" passing from user to kernel.
      "./osprdaccess –r –l –n"
      "./osprdaccess –r –l –n –s 10"
      "./osprdaccess –r –l –n –S 10 20"
      "./osprdaccess –r –l –n –b 10"
      "./osprdaccess –r –l –n –B 10 20";

```
debian:/tmp/cs111/lab2-shijian# ./osprdaccess -r -l -n
Byte_begin: 0  Byte_end: 16383
debian:/tmp/cs111/lab2-shijian# ./osprdaccess -r -l -n -s 10
Byte_begin: 5120  Byte_end: 5631
debian:/tmp/cs111/lab2-shijian# ./osprdaccess -r -l -n -S 10 20
Byte_begin: 5120  Byte_end: 10751
debian:/tmp/cs111/lab2-shijian# ./osprdaccess -r -l -n -b 10
Byte_begin: 10  Byte_end: 10
debian:/tmp/cs111/lab2-shijian# ./osprdaccess -r -l -n -B 10 20
Byte_begin: 10  Byte_end: 20
```

   b) Test case for checking the variables "offset" and "size" passing from user to kernel.
      "echo abcd | ./osprdaccess –w –l"
      "echo abcdefg | ./osprdaccess –o 5 –w –l"
      "echo abcdefg | ./osprdaccess –o 10 –w 3 –l";

```
debian:/tmp/cs111/lab2-shijian# echo abcd | ./osprdaccess -w -l
Write_begin: 0  Write_end: 4
debian:/tmp/cs111/lab2-shijian# echo abcdefg | ./osprdaccess -o 5 -w -l
Write_begin: 5  Write_end: 12
debian:/tmp/cs111/lab2-shijian# echo abcdefg | ./osprdaccess -o 10 -w 3 -l
Write_begin: 10  Write_end: 12
```

4) Process is notified when specific area of ramdisk is modified. (Part 2)
   a) Request a notification for specific sector.
      "./osprdaccess –o 1024 –r –l –n –s 2 &"
      "echo aaa | ./osprdaccess –w –l –d 0.2 &"
      "echo bbb | ./osprdaccess –o 1050 –w –l –d 0.4"

```
debian:/tmp/cs111/lab2-shijian# ./osprdaccess -o 1024 -r -l -n -s 2 &
[2] 3190
debian:/tmp/cs111/lab2-shijian# echo aaa | ./osprdaccess -w -l -d 0.2 &
[3] 3192
debian:/tmp/cs111/lab2-shijian# echo bbb | ./osprdaccess -o 1050 -w -l -d 0.4
Process[3190] received the notification!
[3]-  Done                    echo aaa | ./osprdaccess -w -l -d 0.2
bbb
```

   b) Request a notification for sector range.
      "./osprdaccess –o 512 –r –l –n –S 1 2 &"
      "echo aaa | ./osprdaccess –w –l –d 0.2 &"
      "echo bbb | ./osprdaccess –o 1534 –w –l –d 0.4"

```
debian:/tmp/cs111/lab2-shijian# ./osprdaccess -o 512 -r -l -n -S 1 2 &
[2] 3216
debian:/tmp/cs111/lab2-shijian# echo aaa | ./osprdaccess -w -l -d 0.2 &
[3] 3218
debian:/tmp/cs111/lab2-shijian# echo bbb | ./osprdaccess -o 1534 -w -l -d 0.4
[3]-  Done                    echo aaa | ./osprdaccess -w -l -d 0.2
debian:/tmp/cs111/lab2-shijian# Process[3216] received the notification!
bbb
```

c) Request a notification for specific byte.

"./osprdaccess –o 90 –r 20 –l –n –b 100 &"

"echo aaa | ./osprdaccess –o 200 –w –l –d 0.2 &"

"echo bbb | ./osprdaccess –o 99 –w –l –d 0.4"

```
debian:/tmp/cs111/lab2-shijian# ./osprdaccess -o 90 -r 20 -l -n -b 100 &
[2] 3222
debian:/tmp/cs111/lab2-shijian# echo aaa | ./osprdaccess -o 200 -w -l -d 0.2 &
[3] 3224
debian:/tmp/cs111/lab2-shijian# echo bbb | ./osprdaccess -o 99 -w -l -d 0.4
[3]-  Done                    echo aaa | ./osprdaccess -o 200 -w -l -d 0.2
debian:/tmp/cs111/lab2-shijian# Process[3222] received the notification!
bbb
```

d) Request a notification for byte range.

"./osprdaccess –o 512 –r –l –n –B 600 700 &"

"echo aaa | ./osprdaccess –o 300 –w –l –d 0.2 &"

"echo bbb | ./osprdaccess –o 598 –w –l –d 0.4"

```
debian:/tmp/cs111/lab2-shijian# ./osprdaccess -o 512 -r -l -n -B 600 700 &
[2] 3207
debian:/tmp/cs111/lab2-shijian# echo aaa | ./osprdaccess -o 300 -w -l -d 0.2 &
[3] 3209
debian:/tmp/cs111/lab2-shijian# echo bbb | ./osprdaccess -o 598 -w -l -d 0.4
Process[3207] received the notification!
bbb
```

5) Solve the ordering issue.

a) "./osprdaccess –r –l –n &"

"./osprdaccess –r –l –n &"

"./osprdaccess –r –l –n &"

"echo aaa | ./osprdaccess –w –l"

```
debian:/tmp/cs111/lab2-shijian# ./osprdaccess -r -l -n &
[1] 3183
debian:/tmp/cs111/lab2-shijian# ./osprdaccess -r -l -n &
[2] 3184
debian:/tmp/cs111/lab2-shijian# ./osprdaccess -r -l -n &
[3] 3185
debian:/tmp/cs111/lab2-shijian# echo aaa | ./osprdaccess -w -l
Process[3183] received the notification!
aaa
Process[3184] received the notification!
aaa
[1]   Done                    ./osprdaccess -r -l -n
[2]-  Done                    ./osprdaccess -r -l -n
debian:/tmp/cs111/lab2-shijian# Process[3185] received the notification!
aaa
```

b) "echo aaa | ./osprdaccess –w –l –n &"

"echo bbb | ./osprdaccess –w –l –n &"

"./osprdaccess –r –l –n &"

"echo ccc | ./osprdaccess –w –l"

```
debian:/tmp/cs111/lab2-shijian# echo aaa | ./osprdaccess -w -l -n &
[1] 3178
debian:/tmp/cs111/lab2-shijian# echo bbb | ./osprdaccess -w -l -n &
[2] 3180
debian:/tmp/cs111/lab2-shijian# ./osprdaccess -r -l -n &
[3] 3181
debian:/tmp/cs111/lab2-shijian# echo ccc | ./osprdaccess -w -l
Process[3178] received the notification!
Process[3180] received the notification!
Process[3181] received the notification!
bbb
```

6) Solve the issue that process is killed while waiting for notification.
   c) "echo aaa | ./osprdaccess –w –l –n &"
   "echo bbb | ./osprdaccess –w –l –n &"
   "./osprdaccess –r –l –n &"
   "kill -9 xxxx &" (xxxx is the pid of second process)
   "echo ccc | ./osprdaccess –w –l"

```
debian:/tmp/cs111/lab2-shijian# echo aaa | ./osprdaccess -w -l -n &
[1] 3170
debian:/tmp/cs111/lab2-shijian# echo bbb | ./osprdaccess -w -l -n &
[2] 3172
debian:/tmp/cs111/lab2-shijian# ./osprdaccess -r -l -n &
[3] 3173
debian:/tmp/cs111/lab2-shijian# kill -9 3172 &
[4] 3174
debian:/tmp/cs111/lab2-shijian# echo ccc | ./osprdaccess -w -l
Process[3170] received the notification!
Process[3173] received the notification!
[2]   Killed                  echo bbb | ./osprdaccess -w -l -n
[4]+  Done                    kill -9 3172
debian:/tmp/cs111/lab2-shijian# aaa
```

4. Summary:

All the implementation results satisfy my expectation. At the beginning of this design problem, I have tried a lot of methods to pass the variables from user space to kernel space. Finally, I choose to use the "arg" parameter in "ioctl" function. However, what I pass to kernel is a pointer, which means that I can pass more than one variable to kernel once a time. When I am passing a pointer to kernel, I need to make sure the pointer is "safe". Otherwise the kernel can collapse due to the invalid pointer. But in my kernel program, I do not implement a function to check the safety of this pointer. Therefore, it may cause some problems.

This design problem actually helps me learn a lot about the kernel programming and how ramdisk works when we are using "read" and "write" function.