OPERATING SYSTEMS

UCCD 2103

OCT 2023

ASSIGNMENT REPORT

Part B

|  |  |  |  |
| --- | --- | --- | --- |
| No. | Student ID | Name | Programme |
| 1 | 2104199 | Teng Zhi Kwang | CS |
| 2 | 2105493 | Ng Jun Yuan | CS |
| 3 | 2104300 | Teh Yee Jie | CS |
| 4 | 2101990 | Tan Yi Xin | CS |

**Task 1: Configure storage quota**

1. Install quota and quotatool.

sudo apt-get update

sudo apt-get install quota quotatool

1. Create a primary partition and format it with ext4 filesystem:

Use gdisk or fdisk to create a new partition with GPT partition table.

sudo gdisk /dev/sdc

n

ENTER

ENTER

ENTER

8300

W

Format the partition with ext4 filesystem.

sudo mkfs.ext4 /dev/sdc1

1. Create the directory for /mnt/data/student199

mkdir /mnt/data/student199

1. Edit /etc/fstab file.

sudo nano /etc/fstab

add a line in /etc/fstab

/dev/sdb1 /mnt/data ext4 defaults,usrquota 0 0

1. Remount /mnt/backup.

sudo mount -a

1. Configure Quota for student199.

Create quota file for /mnt/backup and update user and group quotas:

sudo quotacheck -cug /mnt/data

Activate disk quota on /mnt/backup:

sudo quotaon /mnt/data

Opens the quota file for tester833 to edit the soft and hard limits:

sudo edquota -u student199

1. Configure hard and soft limits for student199.

In the file, change soft limit to 50M and hard limit to 60M.

1. Login as student199 and attempt to copy a file larger than 60MB and a file between 50 and 60 MB.

cp 120MB.bin /mnt/data/student199

cp 55MB.bin /mnt/data/student199

Screenshots

The content of /etc/fstab with the entry for /mnt/data

A computer screen with text and numbers

Description automatically generated

Quota configuration for student199. Refer to Figure 4.

A screenshot of a computer

Description automatically generated

A screenshot that shows a file to be copied to /mnt/data/student199 is larger than 60 MB.

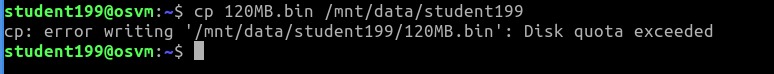
Refer to Figure 6.

A screenshot of a computer

Description automatically generated

A screenshot that shows the error message in attempt to copy a file that exceeds the quota

to /mnt/data/student199. Refer to Figure 7.



**Task 2 Logical Volume Management (LVM) - Configure Logical Volume**

1. Install quota and quotatool.

sudo apt-get update

sudo apt-get install quota quotatool

1. Use the vgcreate command to create a volume group named vg-recovery using the 60 MB hard disk

sudo vgcreate vg-recovery /dev/sdc

1. Use the lvcreate command to create a logical volume named lv-user within the vg-recovery volume group using the entire available space

sudo lvcreate -n lv-user -l 100%FREE vg-recovery

1. Create an ext4 filesystem on the logical volume lv-user.

sudo mkfs.ext4 /dev/vg-recovery/lv-user

1. Find UUIC of lv-user.

sudo blkid

1. Open /etc/fstab file.

sudo nano /etc/fstab

1. Edit /etc/fstab file.

Add a line “UUID=<UUID\_of\_lv-user> /mnt/lvm ext4 defaults,usrquota,grpquota 0 2” into the file.

1. Mount the logical volume to the mount point /mnt/lvm

sudo mkdir /mnt/lvm

sudo mount -o remount /mnt/lvm

1. Configure Quota.

-Create quota file for /mnt/lvm and update user and group quotas:

sudo quotacheck -cug /mnt/lvm

-Activate disk quota on /mnt/lvm:

sudo quotaon /mnt/lvm

-Opens the quota file for tengzhikwang to edit the soft and hard limits:

sudo edquota -u tengzhikwang

1. Configure hard and soft limits for tengzhikwang.
2. Try to copy a file 70MB.bin to the /mnt/lvm directory that exceeds the logical volume size.

sudo cp Downloads/70MB.bin /mnt/lvm

1. Remove 70MB.bin

sudo rm -r /mnt/lvm/70MB.bin

For 100MB hard disk

1. Use fdisk to partition the 100 MB hard disk.

sudo fdisk /dev/sdc

n

ENTER

ENTER

+50M

Repeat another time.

N

ENTER

ENTER

ENTER

w

n

1. Create ext4 filesystems on the partitions

sudo mkfs.ext4 /dev/sdc1

sudo mkfs.ext4 /dev/sdc2

1. Create New Physical Volume, Extend Volume Group with New Physical Volume, Resize the Filesystem

sudo pvcreate /dev/sdd1

sudo vgextend vg-recovery /dev/sdd1

sudo lvextend -l +100%FREE /dev/vg-recovery/lv-user

sudo resize2fs /dev/vg-recovery/lv-user

1. Configure hard and soft limits for tengzhikwang.
2. Try to copy a file 70MB.bin to the /mnt/lvm directory.

sudo cp Downloads/70MB.bin /mnt/lvm

A screenshot that shows two additional hard disks have been added. Refer to Figure 8 as an

example. One is in sdc and one is in sdd.

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

The information of the logical volume created, refer to Figure 9 as an example.

A black screen with white text

Description automatically generated

A screenshot to show the logical volume is mounted to the mount point /mnt/lvm. Refer to Figure 10 as an example.

A black screen with white text

Description automatically generated

A screenshot that shows error writing files to the logical volume. Refer to Figure 11 as an

example.



A screenshot that shows additional partition has been added to volume group.

A screenshot of a computer program

Description automatically generated

Screenshots that show the logical volume has been expanded.

A screenshot of a computer

Description automatically generated

A screenshot that shows the file can be copied successfully.

A screenshot of a computer

Description automatically generated

**Task 3: Synchronisation with semaphores**

Complete C code for mining\_solution.c:

#include <stdio.h>

#include <stdlib.h>

#include <unistd.h>

#include <pthread.h>

#include <semaphore.h>

#include <signal.h>

#define NUM\_EXPLORERS 2

#define NUM\_MINERS 3

#define NUM\_TRANSPORTERS 3

sem\_t explorer\_sem, miner\_sem, transporter\_sem, mining\_sem,transporting\_sem,exploring\_sem;

pthread\_t explorers[NUM\_EXPLORERS], miners[NUM\_MINERS], transporters[NUM\_TRANSPORTERS];

int idle\_miners= NUM\_MINERS, idle\_transporters=NUM\_TRANSPORTERS;

pthread\_mutex\_t miner\_mutex = PTHREAD\_MUTEX\_INITIALIZER;

pthread\_mutex\_t transporter\_mutex = PTHREAD\_MUTEX\_INITIALIZER;

// Signal handler for CTRL+C

void signal\_handler(int signum) {

printf("\n\033[1;97m======================================\033[0m\n");

printf("\n\033[1;97mReceived CTRL+C. Cancel all thread...\033[0m\n");

// Cancel all threads

for (int i = 0; i < NUM\_EXPLORERS; i++){

pthread\_cancel(explorers[i]);

printf("\033[1;97mCancelling Explorer %d\033[0m\n", i+1);

}

for (int i = 0; i < NUM\_MINERS; i++){

pthread\_cancel(miners[i]);

printf("\033[1;97mCancelling Miner %d\033[0m\n", i+1);

}

for (int i = 0; i < NUM\_TRANSPORTERS; i++){

pthread\_cancel(transporters[i]);

printf("\033[1;97mCancelling Transporter %d\033[0m\n", i+1);

}

// Exit

printf("\n\033[1;97mTerminated..\033[0m\n");

exit(0);

}

// Explorer thread function

void \*explorer(void \*arg) {

int id = \*((int \*)arg);

while (1) {

sem\_wait(&explorer\_sem);

printf("\033[1;96mExplorer-%d: exploring...\033[0m\n", id);

sleep(rand() % 3 + 2); // Simulate exploration time

sem\_post(&miner\_sem);

sem\_wait(&explorer\_sem);

sem\_post(&transporting\_sem);

sem\_wait(&exploring\_sem);

sem\_post(&explorer\_sem);

printf("\033[1;96mExplorer-%d: target found!\033[0m\n", id);

sem\_post(&mining\_sem);

}

return NULL;

}

// Miner thread function

void \*miner(void \*arg) {

int id = \*((int \*)arg);

while (1) {

sem\_wait(&miner\_sem);

sleep(0.5);

printf("\033[1;93mMiner-%d: waiting...\033[0m\n", id);

sem\_post(&explorer\_sem);

sem\_wait(&mining\_sem);

idle\_miners--;

printf("\033[1;93mMiner-%d: mining... Idle Miner:%d\033[0m\n", id, idle\_miners);

sleep(rand() % 4 + 2); // Simulate extraction time

printf("\033[1;93mMiner-%d: mined quantity = %d Idle Miner:%d\033[0m\n", id, rand() % 5 + 1, idle\_miners);

idle\_miners++;

sem\_post(&transporter\_sem);

sem\_post(&explorer\_sem);

}

return NULL;

}

// Transporter thread function

void \*transporter(void \*arg) {

int id = \*((int \*)arg);

while (1) {

sleep(2);

sem\_wait(&transporting\_sem);

printf("\033[1;95mTransporter-%d: waiting...\033[0m\n", id);

idle\_transporters--;

sem\_post(&exploring\_sem);

sem\_wait(&transporter\_sem);

printf("\033[1;95mTransporter-%d: transporting... Idle transporter: %d\033[0m\n", id, idle\_transporters);

sleep(4); // Simulate transport time

idle\_transporters++;

sem\_post(&explorer\_sem);

}

return NULL;

}

int main() {

// Initialize semaphores

if (sem\_init(&explorer\_sem, 0, 1) != 0 || sem\_init(&miner\_sem, 0, 3) != 0 || sem\_init(&transporter\_sem, 0, 0) != 0|| sem\_init(&mining\_sem, 0, 0) != 0|| sem\_init(&transporting\_sem, 0, 3) != 0|| sem\_init(&exploring\_sem, 0, 0) != 0) {

perror("Semaphore initialization failed");

exit(EXIT\_FAILURE);

}

// Register signal handler for CTRL+C

signal(SIGINT, signal\_handler);

// Create explorer threads

for (int i = 0; i < NUM\_EXPLORERS; i++) {

int \*id = malloc(sizeof(int));

\*id = i + 1;

if (pthread\_create(&explorers[i], NULL, explorer, id) != 0) {

perror("Thread creation failed");

exit(EXIT\_FAILURE);

}

}

// Create miner threads

for (int i = 0; i < NUM\_MINERS; i++) {

int \*id = malloc(sizeof(int));

\*id = i + 1;

if (pthread\_create(&miners[i], NULL, miner, id) != 0) {

perror("Thread creation failed");

exit(EXIT\_FAILURE);

}

}

// Create transporter threads

for (int i = 0; i < NUM\_TRANSPORTERS; i++) {

int \*id = malloc(sizeof(int));

\*id = i + 1;

if (pthread\_create(&transporters[i], NULL, transporter, id) != 0) {

perror("Thread creation failed");

exit(EXIT\_FAILURE);

}

}

// Wait for threads to finish (should never happen)

for (int i = 0; i < NUM\_EXPLORERS; i++)

pthread\_join(explorers[i], NULL);

for (int i = 0; i < NUM\_MINERS; i++)

pthread\_join(miners[i], NULL);

for (int i = 0; i < NUM\_TRANSPORTERS; i++)

pthread\_join(transporters[i], NULL);

// Destroy semaphores

sem\_destroy(&explorer\_sem);

sem\_destroy(&miner\_sem);

sem\_destroy(&transporter\_sem);

sem\_destroy(&transporting\_sem);

sem\_destroy(&mining\_sem);

sem\_destroy(&exploring\_sem);

return 0;

}

Screenshot that shows the output of the execution of the program

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated