מקבול תוכניות

כעת, בחלק זה נבצע מנגנון שמירה ושחזור של מצב המחסן לקובץ.

מטרתנו הינה לאפשר לשרת לשמור את המצב הפנימי של המחסן (משמע הכמויות של סוגי האטומים שיש לנו) לתוך קובץ ← ולשחזר ממנו את המצב כאשר השרת מופעל מחדש.

הוספנו תמיכה בפרמטר f , אשר מציין את הנתיב של קובץ לשמירת מצב האטומים, כך למשל זה אמור להראות:

./atom_warehouse -f /tmp/state.bin

עשינו זאת ב-2 הקבצים של הלקוחות

```
switch (opt)
{
    case 'f':
        uds_path=optarg; // Store UDS path
        break;
```

כעת, נגדיר בקובץ השרת שלנו מבנה שמייצג את מצב המחסן, זהו ה-state שנשמר לקובץ:

```
// Structure to hold the stock of atoms
struct Stock
{
    unsigned long long hydrogen;
    unsigned long long oxygen;
    unsigned long long carbon;
};
```

כעת, נוסיף קוד לפתיחת קובץ, מיפוי ל-mmap ונעילה:

אם הקובץ אינו קיים← ניצור אותו , ונכניס לתוכו את הערכים שהוזנו ל-command line.

במידה והקובץ שלנו אכן קיים← נטען ממנו את מצב האטומים שלנו:

```
Stock *stock_ptr = nullptr;
int stock_fd = -1;
bool use_save_file = !save_file_path.empty();

if (use_save_file)
{
    // Open or create the file
    stock_fd = open(save_file_path.c_str(), O_RDWR | O_CREAT, 0666);
    if (stock_fd < 0)
    {
        perror("Failed to open save file");
        return 1;
    }
```

O_RDWR הינו מצב קריאה-כתיבה, ו-O_CREAT הינו מצב שאם הקובץ שלנו אינו קיים אז ניצור אותו עם הרשאות 0666.

נדגיש כי מצב ה-open זה או פותח את הקובץ שלנו (אשר קיים) או יוצר את הקובץ , כל זה לפי דרישות המשתמש.

```
// Check if file exists and is the right size
struct stat st;
bool file_exists = (fstat(stock_fd, &st) == 0 && st.st_size == sizeof(Stock));
if (!file_exists)
{
    // Set file size
    if (ftruncate(stock_fd, sizeof(Stock)) < 0)
    {
        perror("Failed to set save file size");
        close(stock_fd);
        return 1;
    }
}
```

נבדוק אם הקובץ שלנו כבר קיים בגודל הנכון ← בדיוק בגודל של stack, אם לא ←נצטרך להגדיר את גודלו, במידה והקובץ לא קיים או בגודל שגוי← נשנה את גודלו ל-sizeof(stock).

```
    // Map the file to memory
    stock_ptr = (Stock *)mmap(nullptr, sizeof(Stock), PROT_READ | PROT_WRITE, MAP_SHARED, stock_fd, 0);
    if (stock_ptr == MAP_FAILED)
    {
        perror("mmap failed");
        close(stock_fd);
        return 1;
    }

    // If file was just created, initialize values
    if (!file_exists)
    {
        stock_ptr->hydrogen = hydrogen_count;
        stock_ptr->oxygen = oxygen_count;
        stock_ptr->carbon = carbon_count;
        msync(stock_ptr, sizeof(Stock), MS_SYNC);
    }
    else
    {
        // Load from file, ignore command-line initialization
        hydrogen_count = stock_ptr->hydrogen;
        oxygen_count = stock_ptr->oxygen;
        carbon_count = stock_ptr->carbon;
    }
}
```

השורה של ה-**mmap**, הינו המנוע של הכל, כאן ממפים את הקובץ שלנו לזיכרון, כאן בעצן כותבים לקובץ פיזית, בזיכרון.

MAP_SHARED ← משותף בין תהליכים.

PROT_READ ← גישת קריאה

PROT_WRITE ← גישת כתיבה

במידה והקובץ שלנו חדש← נכתוב אליו את הערכים שקיבלנו ב-command line .

במידה והקובץ שלנו כבר היה קיים← קוראים ממנו את הערכים, ומשתמשים בהם.

כעת, בכל שינוי , אשר מתבטא בפעולת ADD או DELIVER , אזי נוסיף את פעולת ה-**flock()** , אשר מבטיח את המצב הבא:

אם פועלים כמה תהליכים במקביל, או שנוצרו כמה מופעים של השרת, אזי ה-**flock()** מבטיח שנעילה תמנע גישה מקבילית, ותשמור על עקביות:

```
flock(stock_fd, LOCK_EX);
stock_ptr->hydrogen = hydrogen_count;
stock_ptr->oxygen = oxygen_count;
stock_ptr->carbon = carbon_count;
msync(stock_ptr, sizeof(Stock), MS_SYNC);
flock(stock_fd, LOCK_UN);
```

הקובץ שלנו נשמר מיד אחרי כל שינוי , זאת כדי לשמר את המידע גם אם השרת שלנו ייסגר, או יקרוס.

ולבסוף, בקובץ השרת נבצע סגירה תקינה של ה-mmap והקובץ :

```
if (use_save_file && stock_ptr)
{
    msync(stock_ptr, sizeof(Stock), MS_SYNC);
    munmap(stock_ptr, sizeof(Stock));
    close(stock_fd);
}
```

כעת, נראה הרצה מסודרת ומקיפה של חלק זה:

נבצע הרצה רגילה רק הפעם עם הדגל של f-, שם גם נכתוב את הקובץ אליו נשמור את המלאי:

```
roy3177@LAPTOP-QCUJB7RP:~/Counting molecules/part6-Concurrency$ ./file_server \
    -T 12345 \
    -U 12345 \
    -s /tmp/stream.sock \
    -d /tmp/dgram.sock \
    -h 10 -o 10 -c 10 \
    -t 80 \
    -f /tmp/stock_state.bin
```

כעת נבצע הרצות רגילות , כמו שעשינו עד חלק 5:

```
roy3177@LAPTOP-QCUJB7RP:~/Counting molecules$ cd part6-Concurrency
roy3177@LAPTOP-QCUJB7RP:~/Counting molecules/part6-Concurrency$ ./atom_supplier 127.0.0.1 12345
Connected to server at 127.0.0.1:12345
Enter command (e.g., ADD HYDROGEN 3 or EXIT): ADD HYDROGEN 3
Server returned: SUCCESS: Atom added successfully.

Enter command (e.g., ADD HYDROGEN 3 or EXIT): ADD OXYGEN 2
Server returned: SUCCESS: Atom added successfully.

Enter command (e.g., ADD HYDROGEN 3 or EXIT): EXIT
Closing connection. Bye!
```

```
roy3177@LAPTOP-QCUJB7RP:~/Counting molecules$ cd part6-Concurrency
roy3177@LAPTOP-QCUJB7RP:~/Counting molecules/part6-Concurrency$ ./molecule_requester 127.0.0.1 12345
Using UDP to 127.0.0.1:12345
Enter command (DELIVER <MOLECULE> <AMOUNT> | EXIT): DELIVER H2O 2
[Server Reply] CANNOT_DELIVER
Enter command (DELIVER <MOLECULE> <AMOUNT> | EXIT): DELIVER WATER 2
[Server Reply] DELIVERED
Enter command (DELIVER <MOLECULE> <AMOUNT> | EXIT): DELIVER ALCOHOL 1
[Server Reply] DELIVERED
Enter command (DELIVER <MOLECULE> <AMOUNT> | EXIT): EXIT
Exiting.
```

```
roy3177@LAPTOP-QCUJB7RP:~/Counting molecules$  cd part6-Concurrency
roy3177@LAPTOP-QCUJB7RP:~/Counting molecules/part6-Concurrency$ ./molecule_requester -f /tmp/dgram.sock

Using UNIX Domain Socket at /tmp/dgram.sock
Enter command (DELIVER <MOLECULE> <AMOUNT> | EXIT): DELIVER WATER 1
[Server Reply] DELIVERED
Enter command (DELIVER <MOLECULE> <AMOUNT> | EXIT): EXIT
Exiting.
```

```
roy3177@LAPTOP-QCUJB7RP:~/Counting molecules$ cd part6-Concurrency
roy3177@LAPTOP-QCUJB7RP:~/Counting molecules/part6-Concurrency$ ./atom_supplier -f /tmp/stream.sock
Connected to server via UDS at /tmp/stream.sock
Enter command (e.g., ADD HYDROGEN 3 or EXIT): ADD OXYEGEN 6
Server returned: ERROR: Unknown atom type. Use HYDROGEN, OXYGEN, or CARBON.

Enter command (e.g., ADD HYDROGEN 3 or EXIT): ADD CARBON 12
Server returned: SUCCESS: Atom added successfully.

Enter command (e.g., ADD HYDROGEN 3 or EXIT): ADD HYDROGEN 7
Server returned: SUCCESS: Atom added successfully.

Enter command (e.g., ADD HYDROGEN 3 or EXIT): EXIT
Closing connection. Bye!
```

כעת, ביצענו הכל והשרת שלנו סיים את החיבור:

```
roy3177@LAPTOP-QCUJB7RP:~/Counting molecules$ cd part6-Concurrency
roy3177@LAPTOP-QCUJB7RP:~/Counting molecules/part6-Concurrency$ ./file_server \
  -T 12345 \
  -U 12345 \
  -s /tmp/stream.sock \
  -d /tmp/dgram.sock \
  -h 10 -o 10 -c 10 \
  -t 80 \
  -f /tmp/stock_state.bin
[INFO] UDS STREAM socket bound to /tmp/stream.sock
[INFO] UDS DATAGRAM socket bound to /tmp/dgram.sock
Server is listening on TCP port 12345 and UDP port 12345...
[INFO] New client connected: 127.0.0.1
[INFO] Atoms: H=13, O=10, C=10
[INFO] Atoms: H=13, O=12, C=10
[INFO] Client disconnected.
[INFO] UDP request received: DELIVER WATER
[INFO] Atoms: H=9, O=10, C=10
[INFO] UDP request received: DELIVER ALCOHOL
[INFO] Atoms: H=3, O=9, C=8
[INFO] UDS DATAGRAM request received: DELIVER WATER 1
[INFO] Atoms: H=1, O=8, C=8
[INFO] New UDS STREAM client connected.
[INFO] Atoms: H=1, O=8, C=20
[INFO] Atoms: H=8, O=8, C=20
[INFO] Client disconnected.
[INFO] Timeout reached. Shutting down.
[INFO] Server shut down.
```

נרצה עכשיו לראות מה המצב של הקובץ שלנו, מה שעבדנו עליו ושמרנו אליו את מחסן האטומים שלנו, לכן נרשום **xxd /tmp/stock_state.bin** , ונקבל את כמויות האטומים לפי הסדר בכתיב של HEX בהקשר של **Little-Endian:**

```
roy3177@LAPTOP-QCUJB7RP:~/Counting molecules/part6-Concurrency$ xxd /tmp/stock_state.bin
00000000: 0800 0000 0000 0000 0800 0000 0000 0000  ................
00000010: 1400 0000 0000 0000                      ........
```

ולכן אפשר לראות כי כמות המימן שנשמרה הינה 8, כמות החמצן שנשמרה הינה8, וכמות הפחמן שנשמרה הינה 20 (כי הרי בכתיב של Little Endian זה 1400 0000 0000 0000)

כעת, נרצה לכתוב לקובץ הזה עוד, כי הרי מה שעשינו בחלק 6 זה לשמור מחסן אטומים שעבדנו עליו, ולהמשיכו משם:

```
roy3177@LAPTOP-QCUJB7RP:~/Counting molecules/part6-Concurrency$ ./file_server \
  -T 12345 \
  -U 12345 \
  -s /tmp/stream.sock \
  -d /tmp/dgram.sock \
  -f /tmp/stock_state.bin \
  -t 60
[INFO] UDS STREAM socket bound to /tmp/stream.sock
[INFO] UDS DATAGRAM socket bound to /tmp/dgram.sock
Server is listening on TCP port 12345 and UDP port 12345...
```

ולכן ,כאשר נכתוב הוספה של עוד אטומים, כמות האטומים מתעדכנת לתוך הקובץ שבו עבדנו קודם:

```
roy3177@LAPTOP-QCUJB7RP:~/Counting molecules/part6-Concurrency$ ./atom_supplier 127.0.0.1 12345
Connected to server at 127.0.0.1:12345
Enter command (e.g., ADD HYDROGEN 3 or EXIT):  ADD HYDROGEN 3
Server returned: SUCCESS: Atom added successfully.

Enter command (e.g., ADD HYDROGEN 3 or EXIT):  ADD OXYGEN 2
Server returned: SUCCESS: Atom added successfully.

Enter command (e.g., ADD HYDROGEN 3 or EXIT): EXIT
Closing connection. Bye!
roy3177@LAPTOP-QCUJB7RP:~/Counting molecules/part6-Concurrency$
```

```
[INFO] New client connected: 127.0.0.1
[INFO] Atoms: H=11, O=8, C=20
[INFO] Atoms: H=11, O=10, C=20
[INFO] Client disconnected.
```

רואים כי כמות האטומים שלנו מתעדכנת בקובץ זה בהתאם !