UNIVERSITE DU QUEBEC A TROIS-RIVIERES

TRAVAIL PRATIQUE 2 (LINUX)

PRESENTE A

FRANCOIS MENIER

COMME EXIGENCE PARTIELLE

DU COURS

SYSTEME D'EXPLOITATION

PAR

OUATTARA ONIFERE (OUAO86070100)

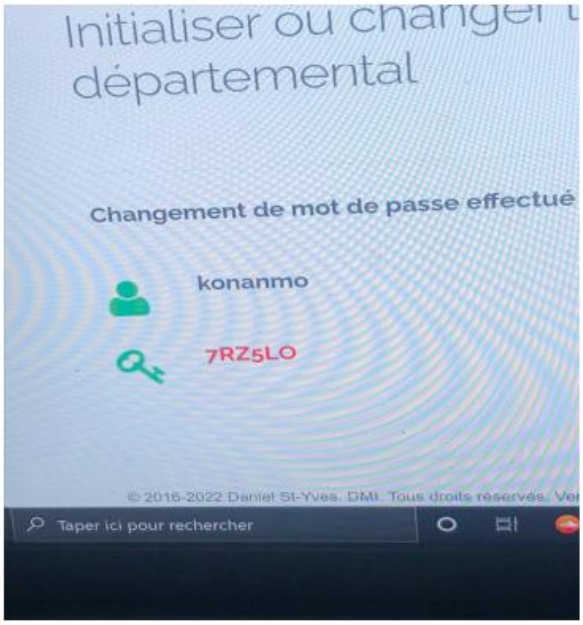
KONAN TRESOR (KONM65510300)

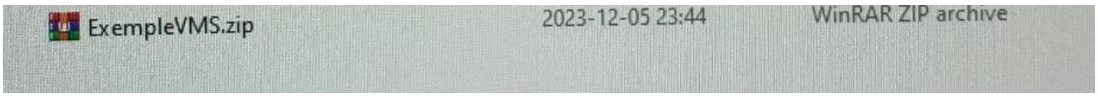
BOWEN NDA MARIE DES ANGES (BOWM7160100)

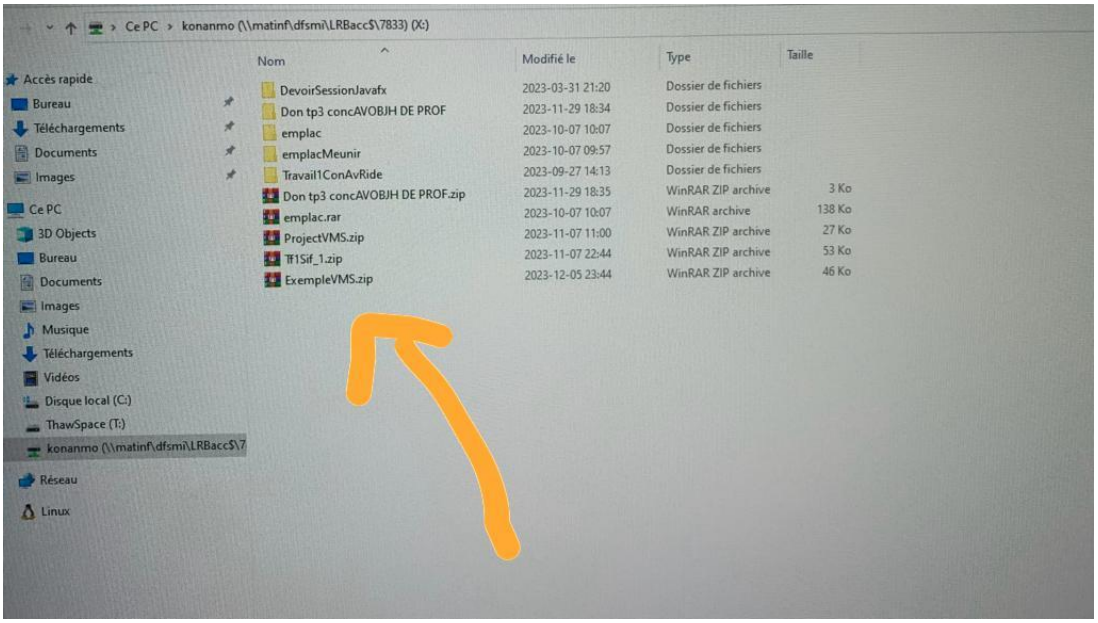
TAMBOURA DJENEBA (TAMD93550100)

AUTOMNE 2023

Liens du travail : X:\







INTRODUCTION

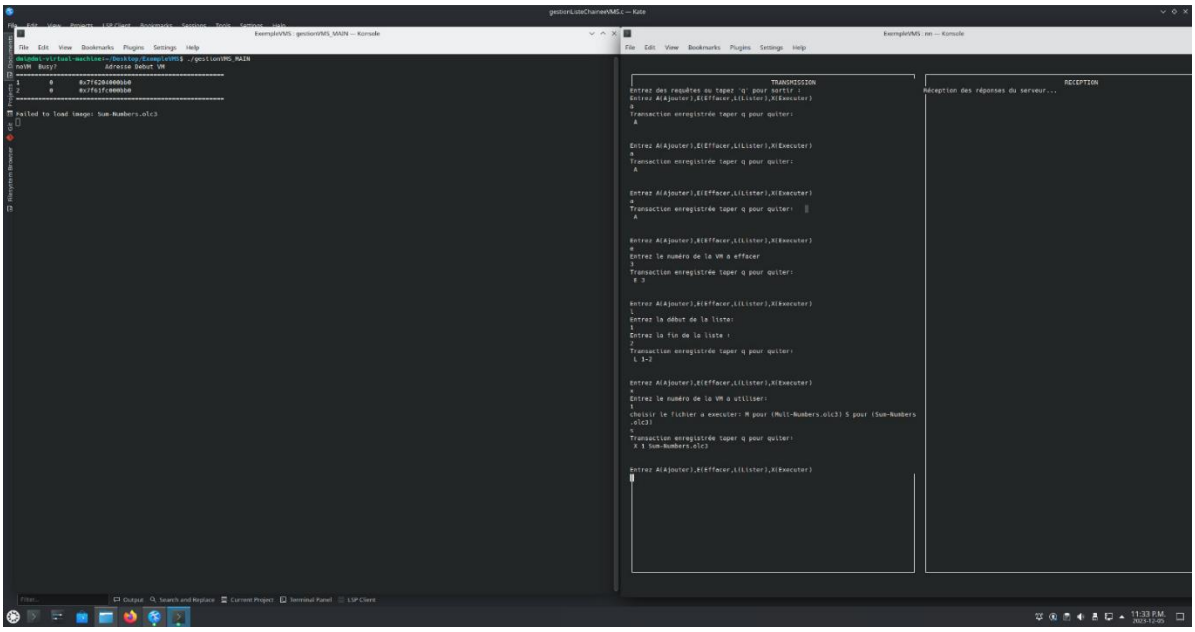
En premier lieu, on s'est inspiré des notions expérimentées dans le programme du TP1, l'utilisation des fenêtres et des concepts de communication entre threads, on doit implémenter un programme permettant la gestion concurrente d'un VMS à l'intérieur d'une architecture Client/Serveur. Des clients pourront alors acheminer des requêtes au serveur FIFO de transmission de transactions, le serveur exécute chacune des transactions et retourne aux clients concernés par une FIFO assignée à chaque client l'information à afficher lors des transactions de listages ou d'exécution.

Mode d'emploi de l'application

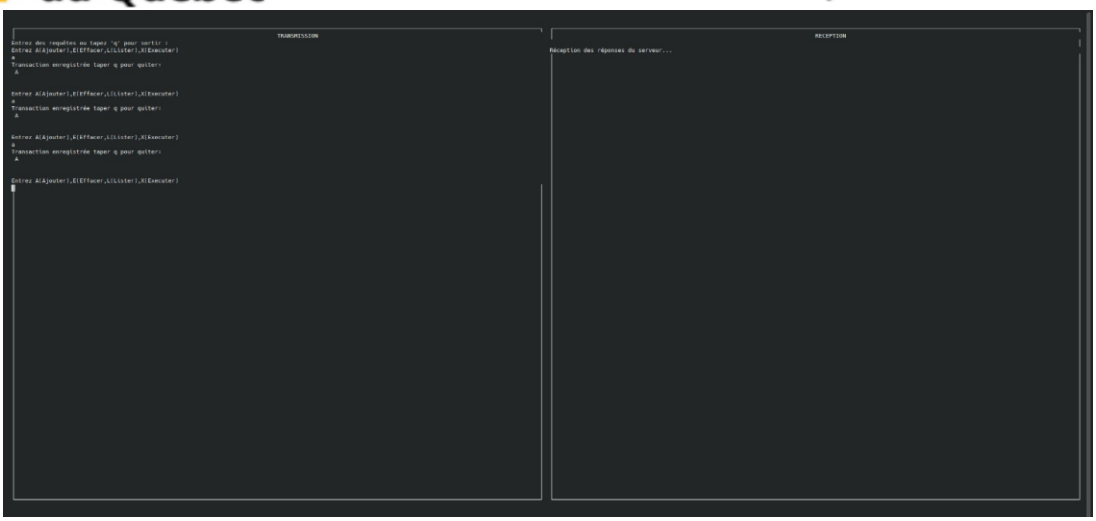
D'abord taper la commande make dans le terminal au répertoire du projet le serveur devra se mettre en marche. Dans un deuxième terminal dans le répertoire du projet compiler le fichier client.c avec la commande gcc -o client Client.c puis ./client pour démarrer les fenêtres Entrer les transactions dans la fenêtre de transmission et taper q une fois termine.

PRÉSENTATION ET EXPLICATION

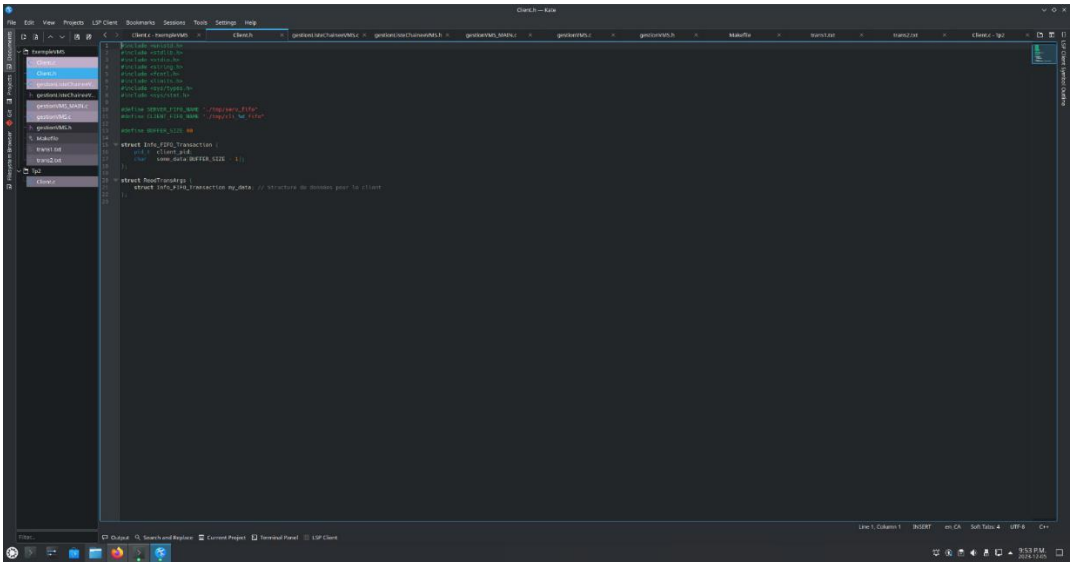
Affichage principal

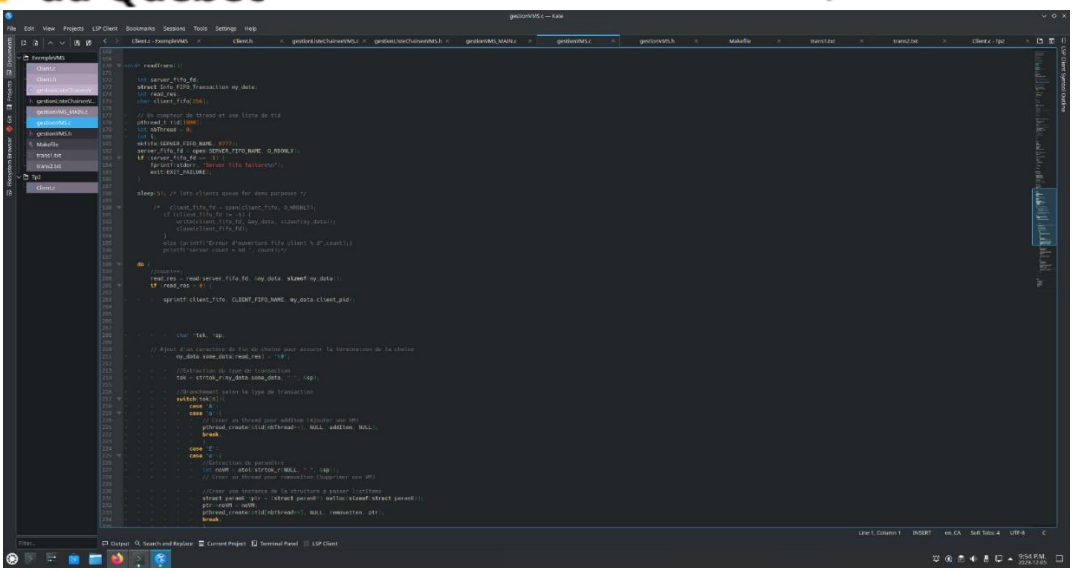


(question a et b)Le programme côté client lors du démarrage ouvre deux fenêtres, une identifiée TRANSMISSION, l'autre RECEPTION. Le programme côté client pourra alors, par la fenêtre de TRANSMISSION, introduire des requêtes permettant d'effectuer des opérations sur une VM noVM donnée. Le programme client achemine ensuite, par exemple, des requêtes d'ajout de VM (transaction A) au serveur et ce par la FIFO de transmission de transactions.
Lors de la fermeture (en tapant quit (ou q) dans la fenêtre de transmission) de l'application Client, les fenêtres côté client sont fermées.

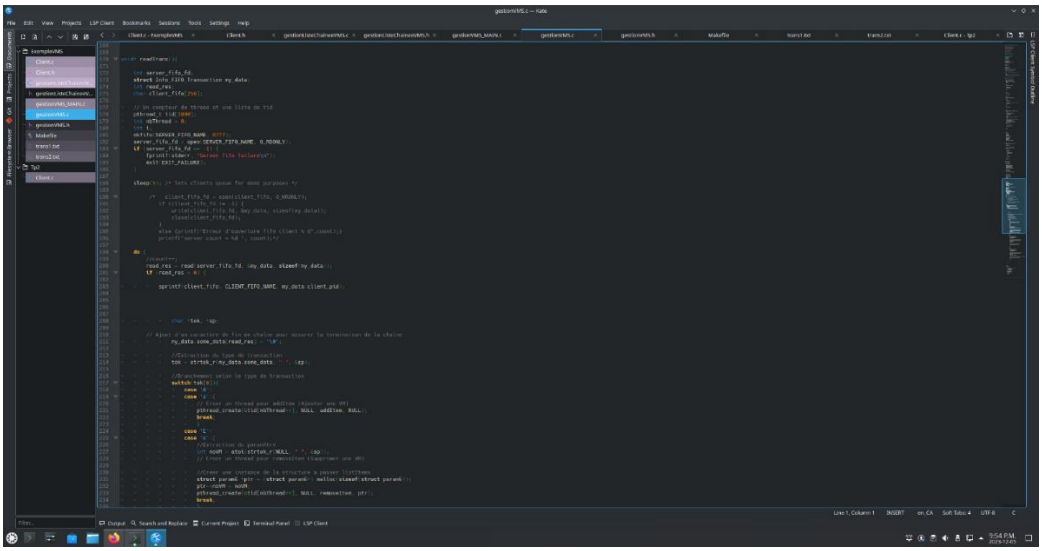


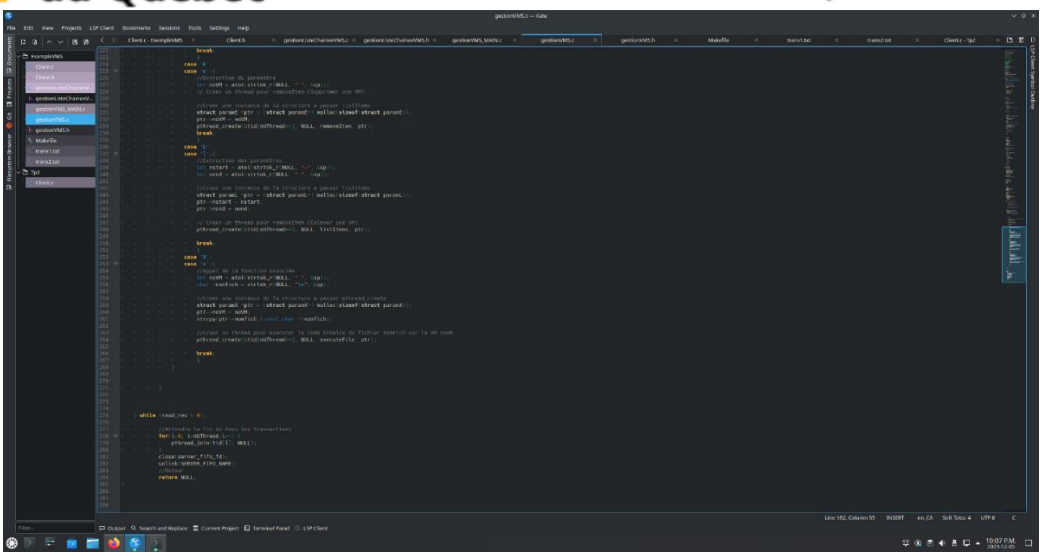
c- La FIFO de transmission de transactions devra s'appeler FIFO_TRANSACTIONS. Cette FIFO est créée par le programme serveur lors de son démarrage et est ouverte par le serveur en lecture (bloquante).





(Question d et e) Ensuite, le serveur (fonction readTrans()) boucle en lecture sur la FIFO FIFO_TRANSACTIONS pour lire les informations provenant des clients.



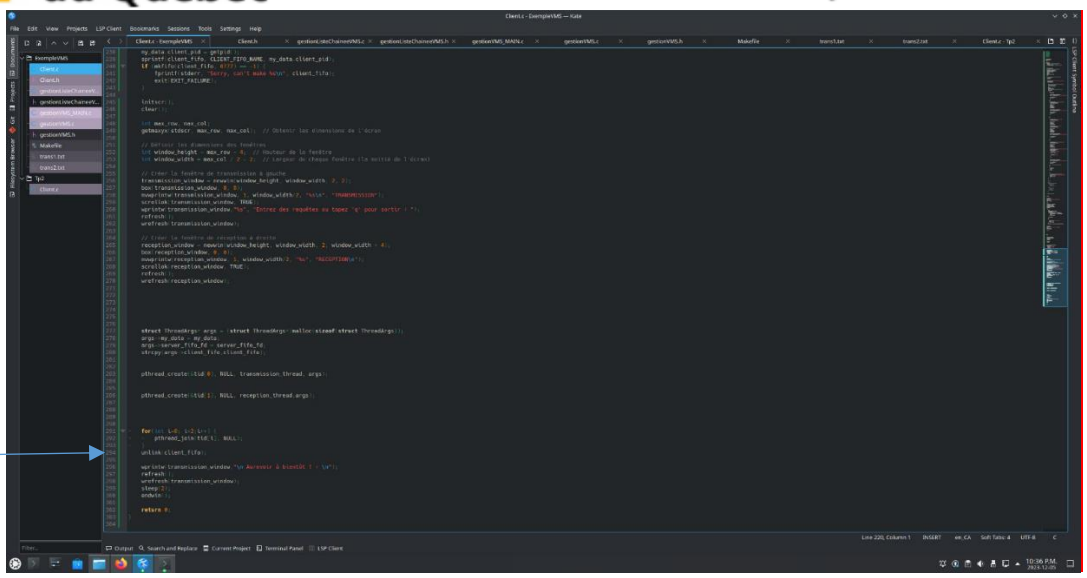


f- Les threads du côté serveur doivent aussi répondre aux clients dans le cas d'une requête d'affichage ou les threads (côté serveur) déposent les structures infoVM sérialisées chacune dans une chaîne de caractères (dimension 400) déposées ensuite dans la FIFO (client_fifo) reliant le serveur et un client particulier qui sont à afficher du côté client.

g- Les threads du côté serveur doivent aussi répondre aux clients dans le cas d'une requête d'exécution ou les threads (côté serveur) déposent les affichages produits par la fonction threadée executeFile() sérialisées dans une chaîne de caractères (dimension 400) déposées ensuite dans la FIFO (client_fifo) reliant le serveur et un client particulier, qui sont à afficher du côté client.

h- Les threads du côté serveur doivent aussi dans le cas de transactions de transmission d'information transmettre le message inséré d'abord dans une chaîne de caractères (dimension 400) et ensuite déposé dans la FIFO (client_fifo) reliant le serveur à chaque client récipiendaire du message particulier qui sont à afficher du côté client.

i- Quand un client décide de terminer la transmission de transactions il doit détruire la FIFO client_fifo (unlink())



j- Du côté client les E/S doivent s'effectuer chacune à l'intérieur d'une fenêtre. L'introduction des transactions se fait donc de façon manuelle à l'intérieur d'une fenêtre du côté client (TRANSMISSION). Les réponses provenant du serveur sont affichées à l'intérieur d'une autre fenêtre (RECEPTION) en caractères d'une couleur différente pour distinguer les réponses découlant de listage et celles de transmission de messages.

k- Du côté client vous devriez utiliser des threads de façon à ce que la lecture des transactions et l'affichage des réponses reçues du serveur soient indépendants.

CONCLUSION

Dans notre travail 2, on a compris à l'aide des connaissances du travail 1 comment implémenter un programme permettant la gestion concurrente d'un VMS à l'intérieur d'une architecture Client/Serveur. A partir de ce travail, on a pu améliorer nos connaissances dans la gestion concurrente d'in VMS.