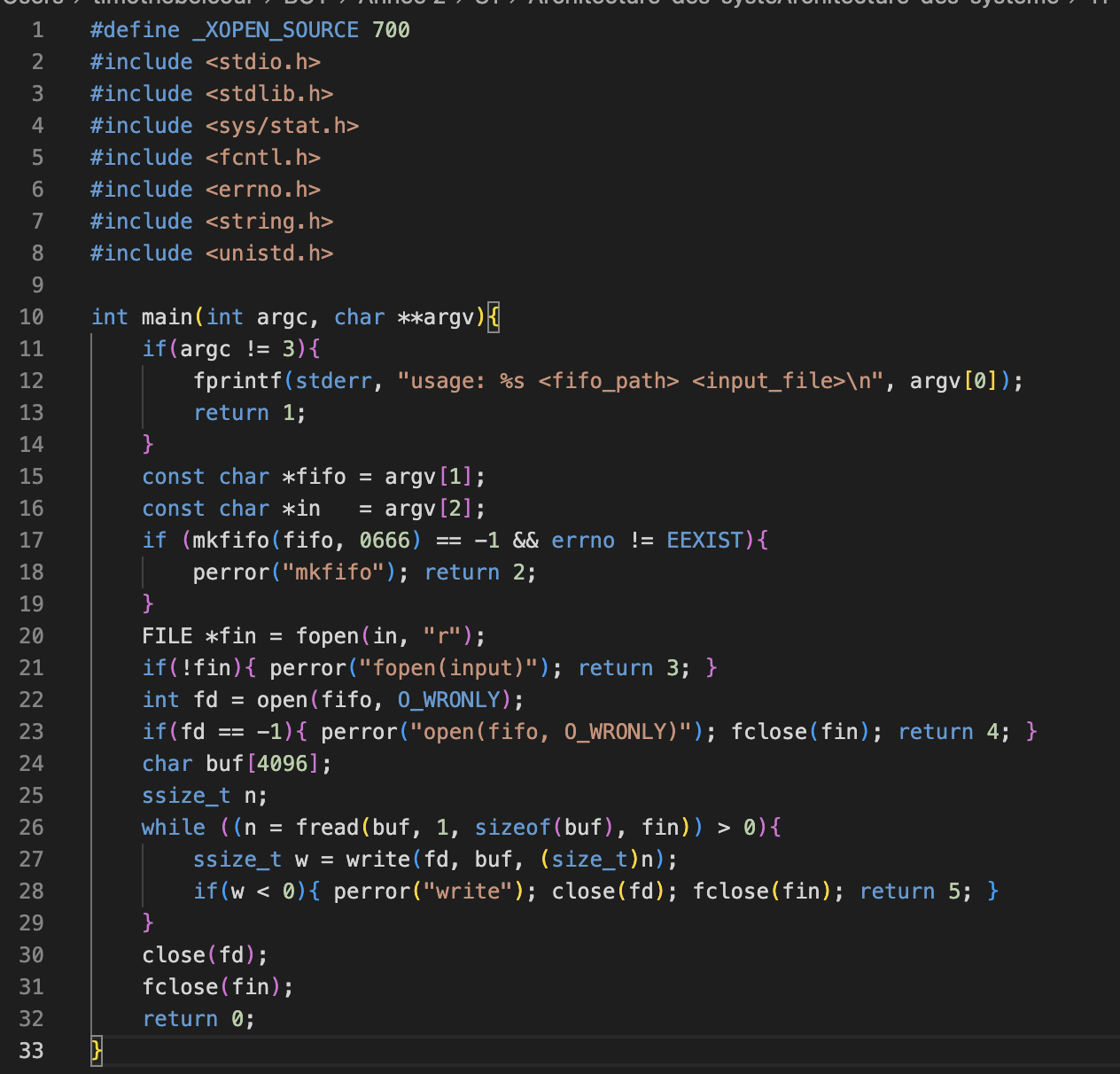
TP UNIX N°1

BUT - INFO 2°Année

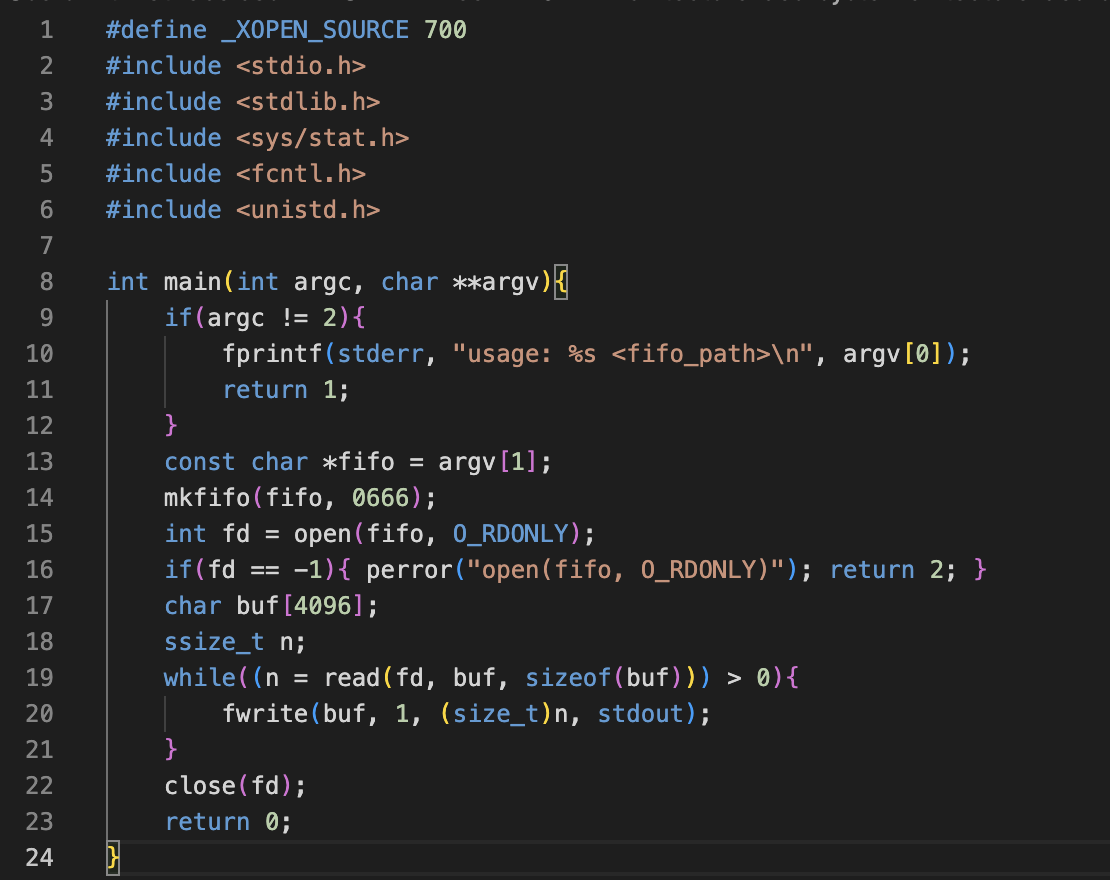
1\_ FIFO

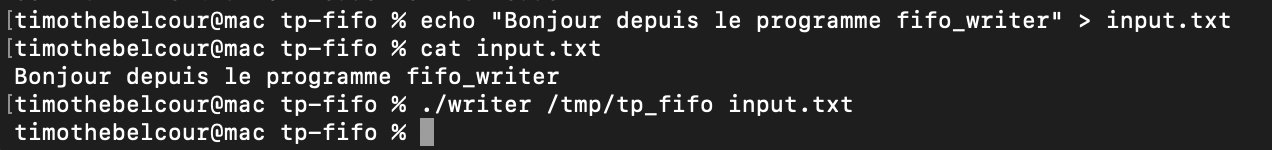
En vous aidant du manuel (commande : man), créez deux programme C permettant d’échanger des messages contenus dans un fichier (donné en paramètre à l’un des deux processus) à travers une FIFO. Le premier processus créé la FIFO et ouvre le fichier en lecture pour lire les données et les écrire dans la FIFO. Le second, lit dans la FIFO et affiche à l’écran les données.

Writer.c :

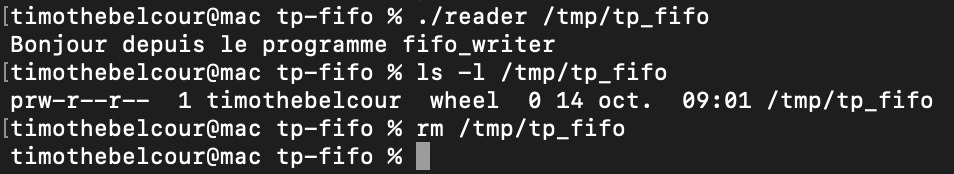


Reader.c :





Writer : crée la FIFO si besoin, ouvre le fichier en lecture, et écrit son contenu dans la FIFO.

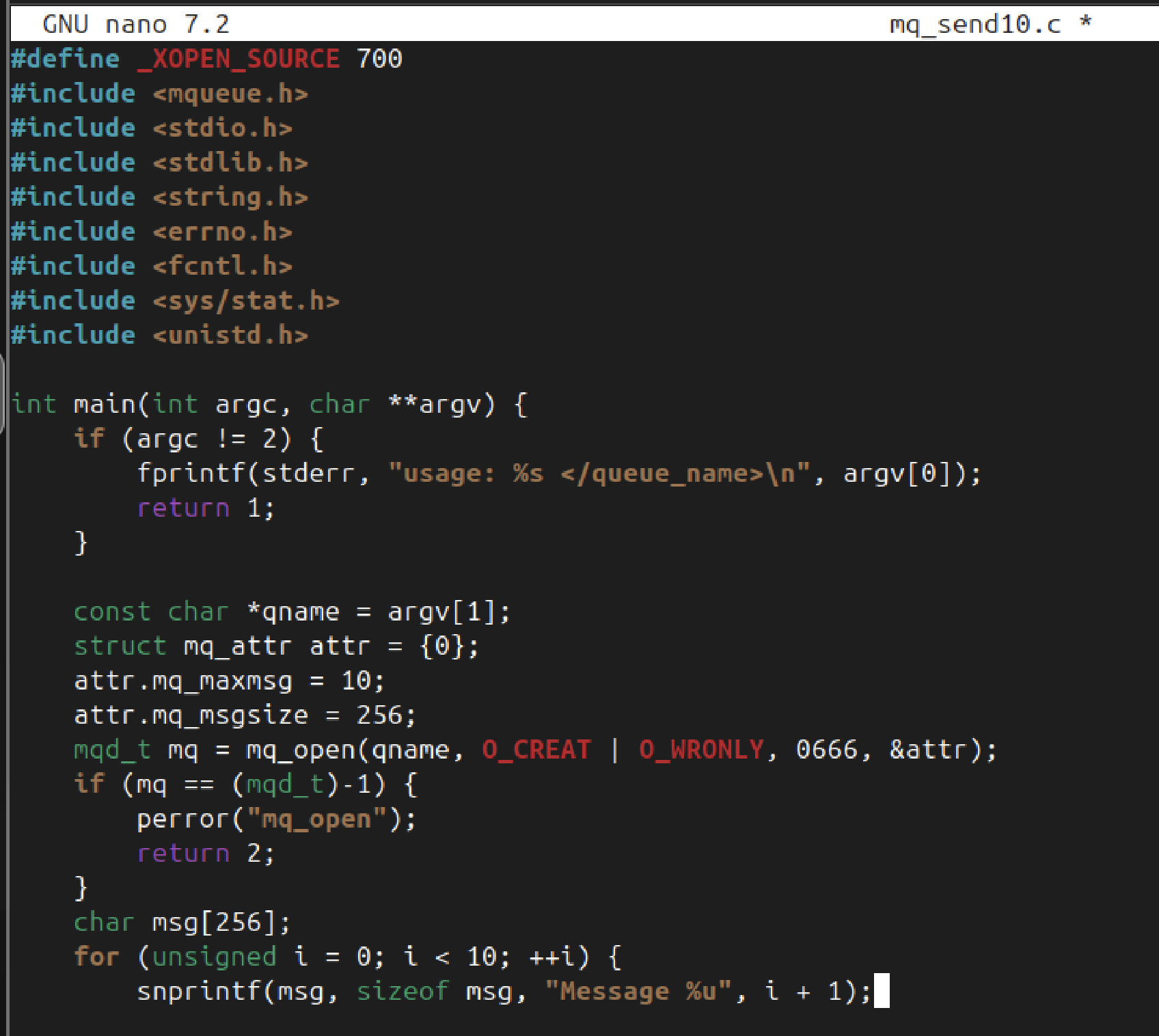


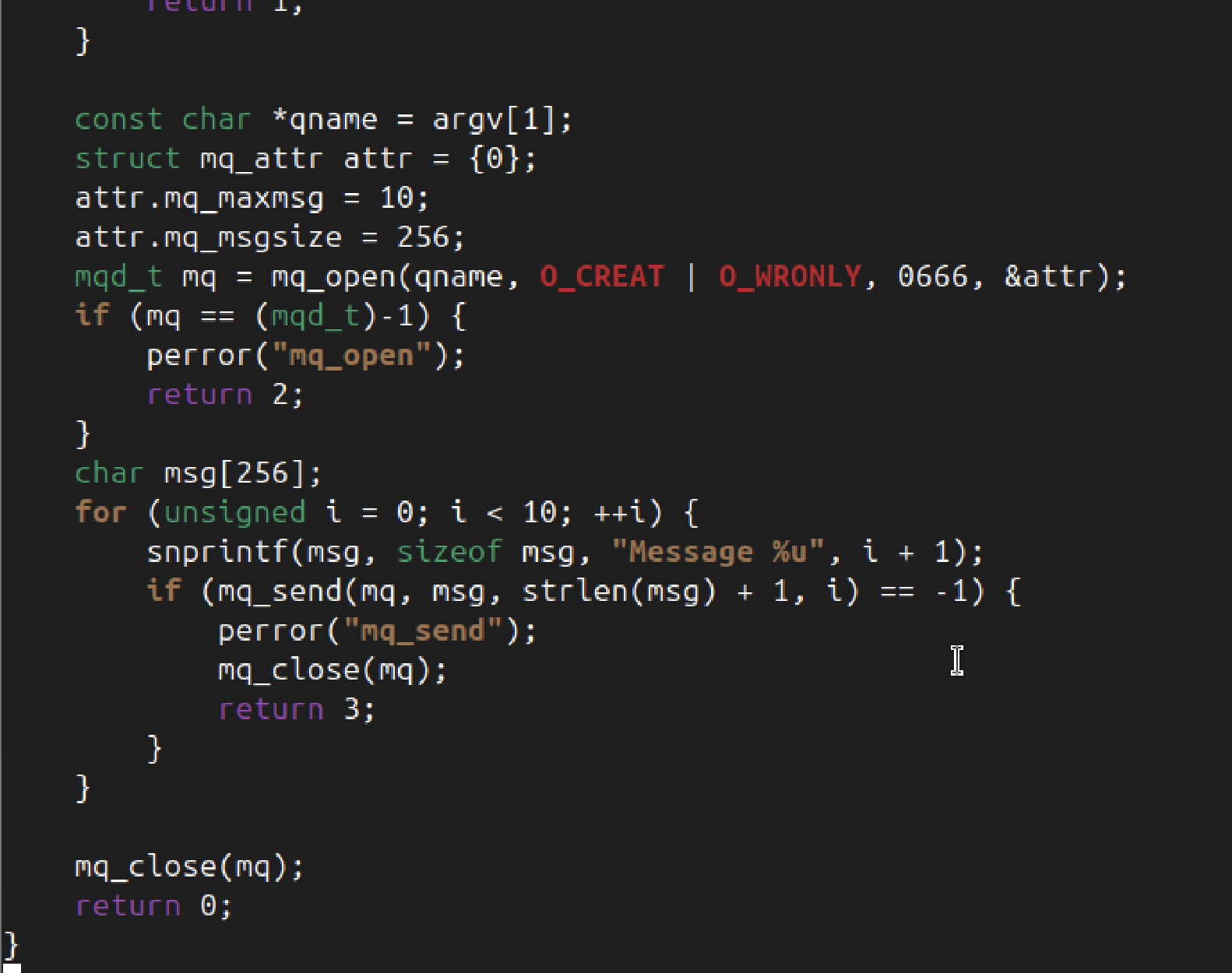
Reader : ouvre la FIFO en lecture et affiche tout ce qu’il reçoit.

Le Terminal A affiche exactement le contenu de input.txt. La FIFO /tmp/tp\_fifo est visible avec ls -l.

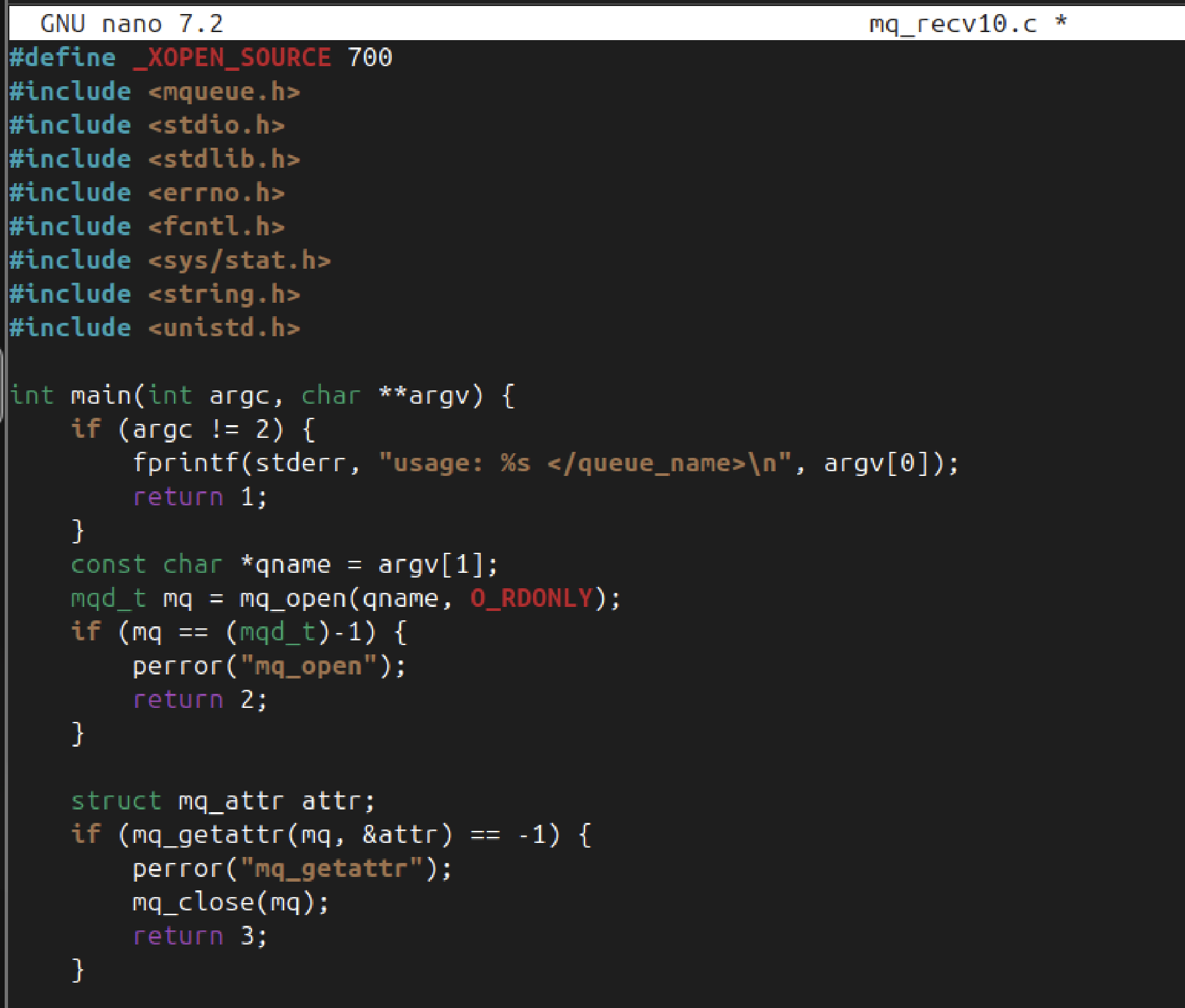
2\_ Les queues de message

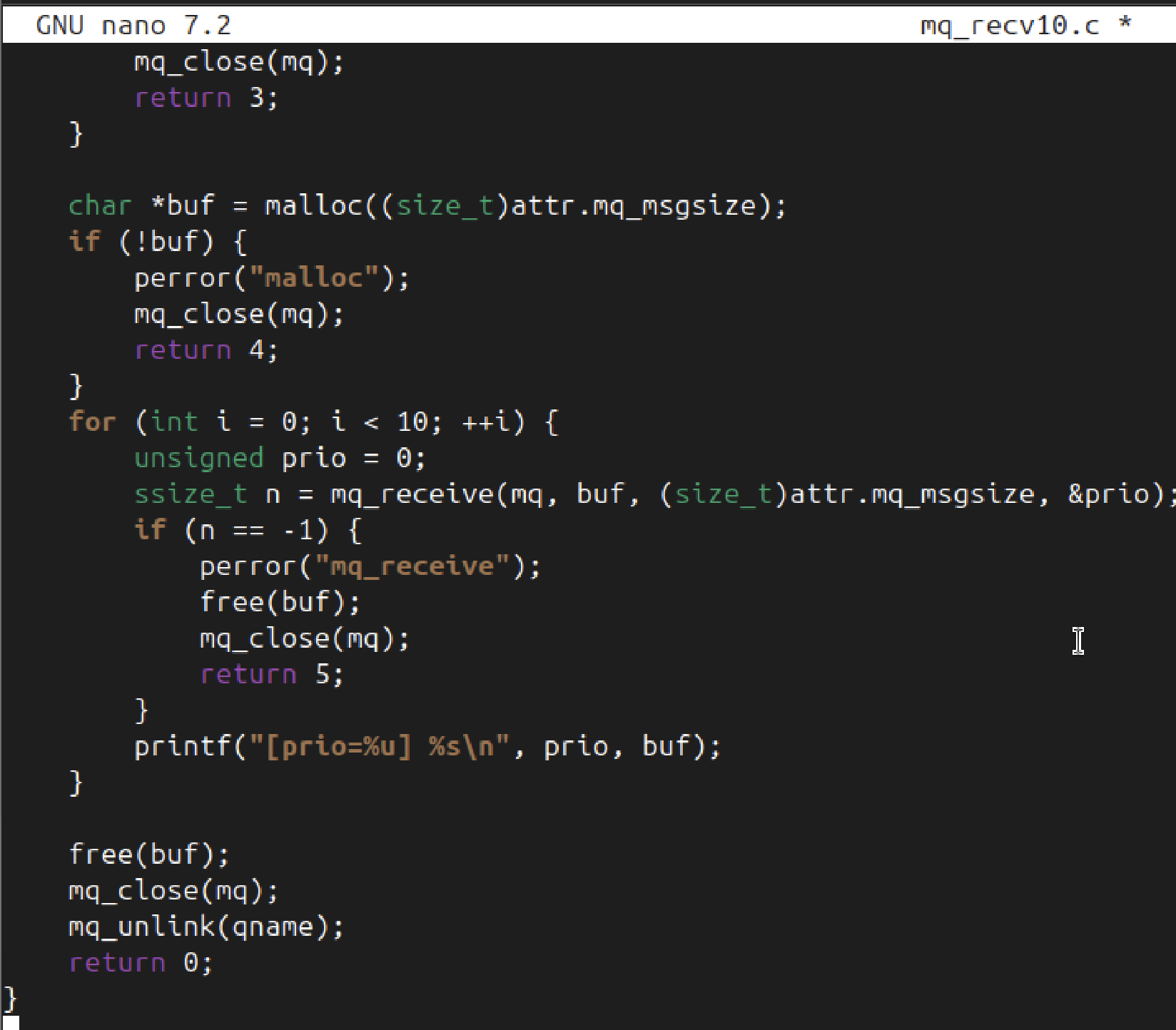
mq\_send10.c :



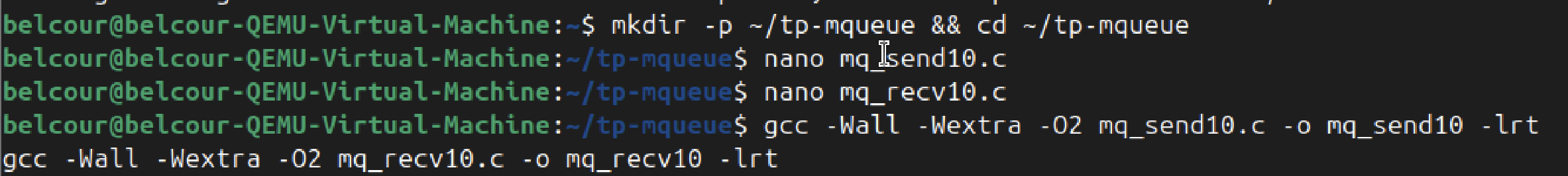


mq\_recv10.c :





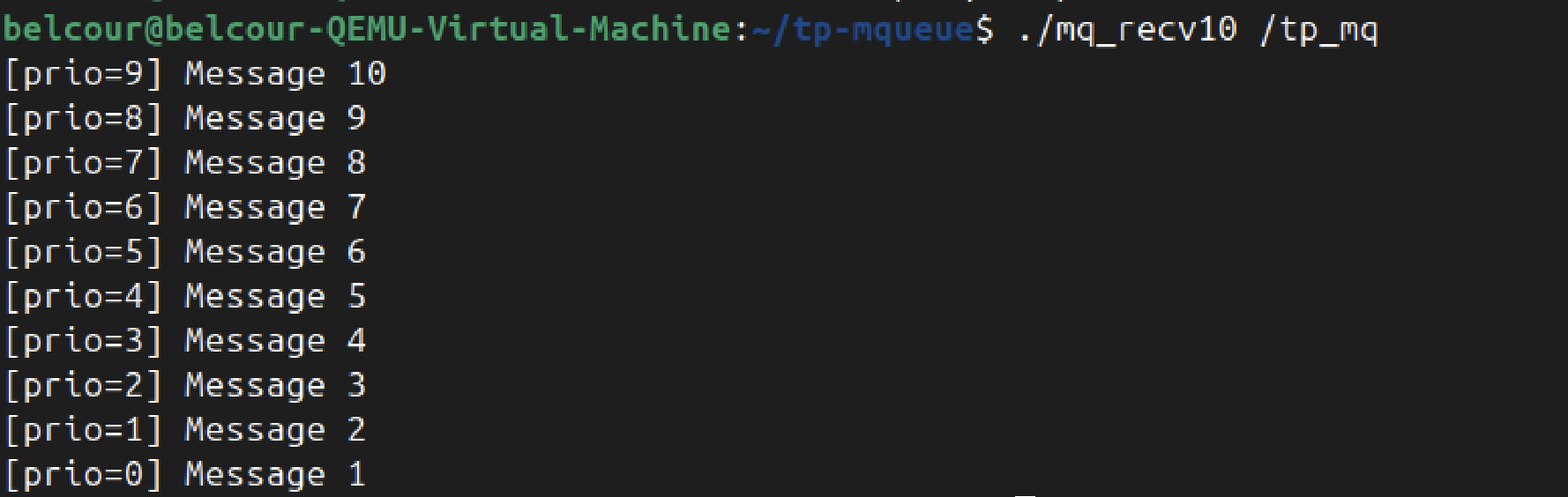
1er Terminal :





mq\_send10.c crée la file de messages /tp\_mq et envoie 10 messages de priorités différentes (0 à 9).

2ème Terminal :

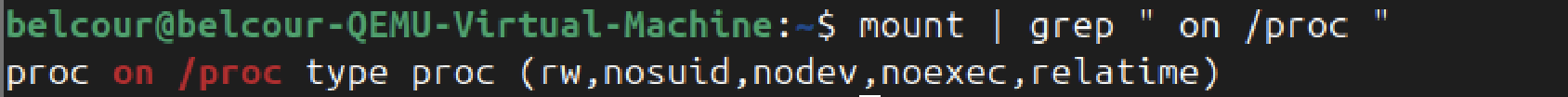
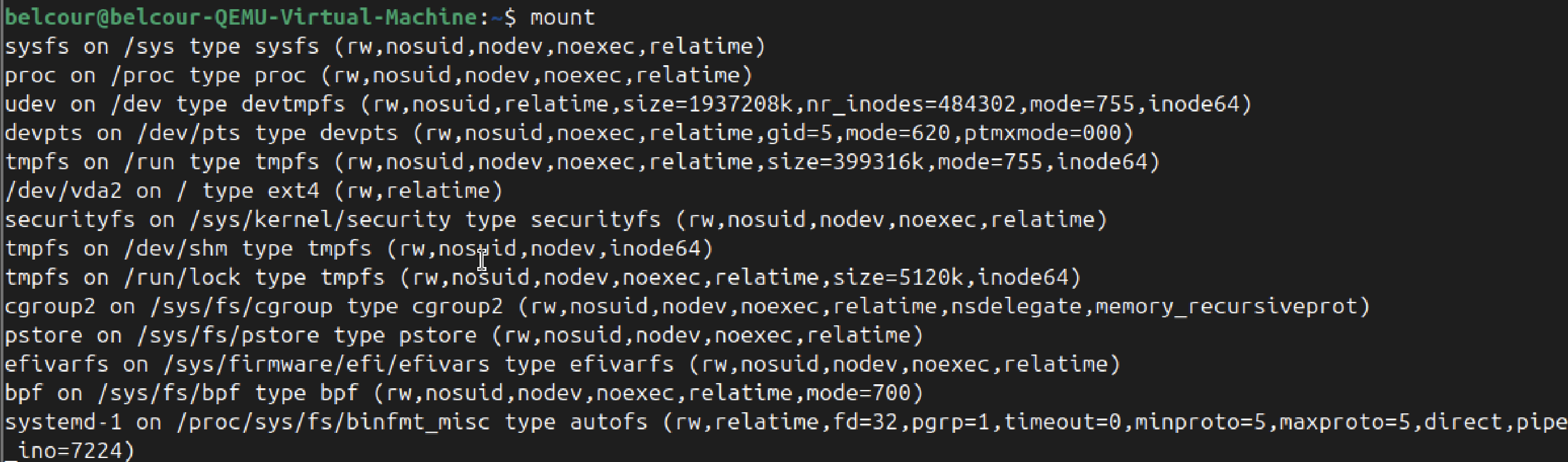


mq\_recv10.c ouvre la file, lit les 10 messages et les affiche avec leur priorité.

La file est ensuite supprimée automatiquement (mq\_unlink) à la fin du programme.

L’affichage montre que la réception se fait selon la priorité décroissante, prouvant le bon fonctionnement de l’API POSIX mqueue.

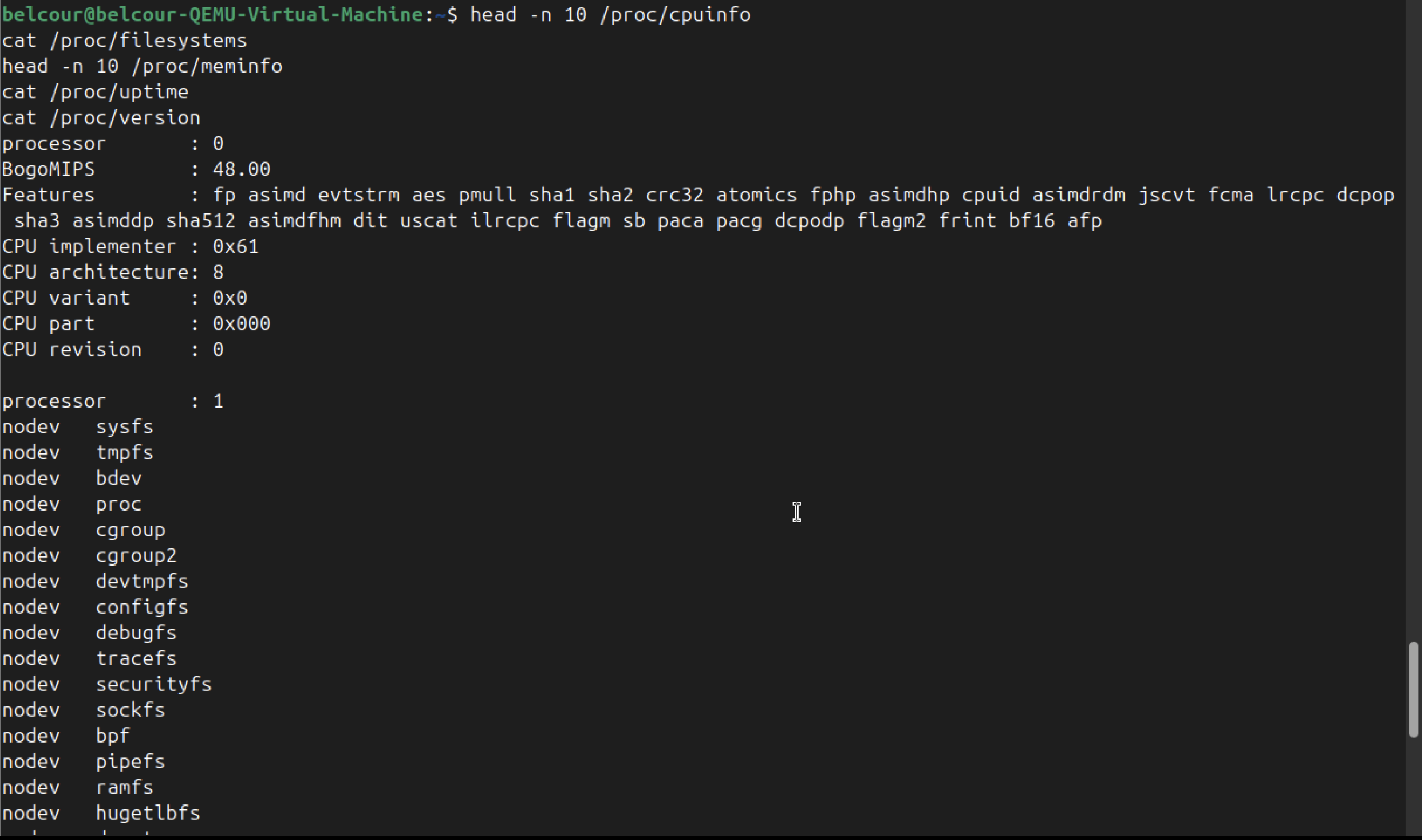
3\_ Le système de fichiers /proc

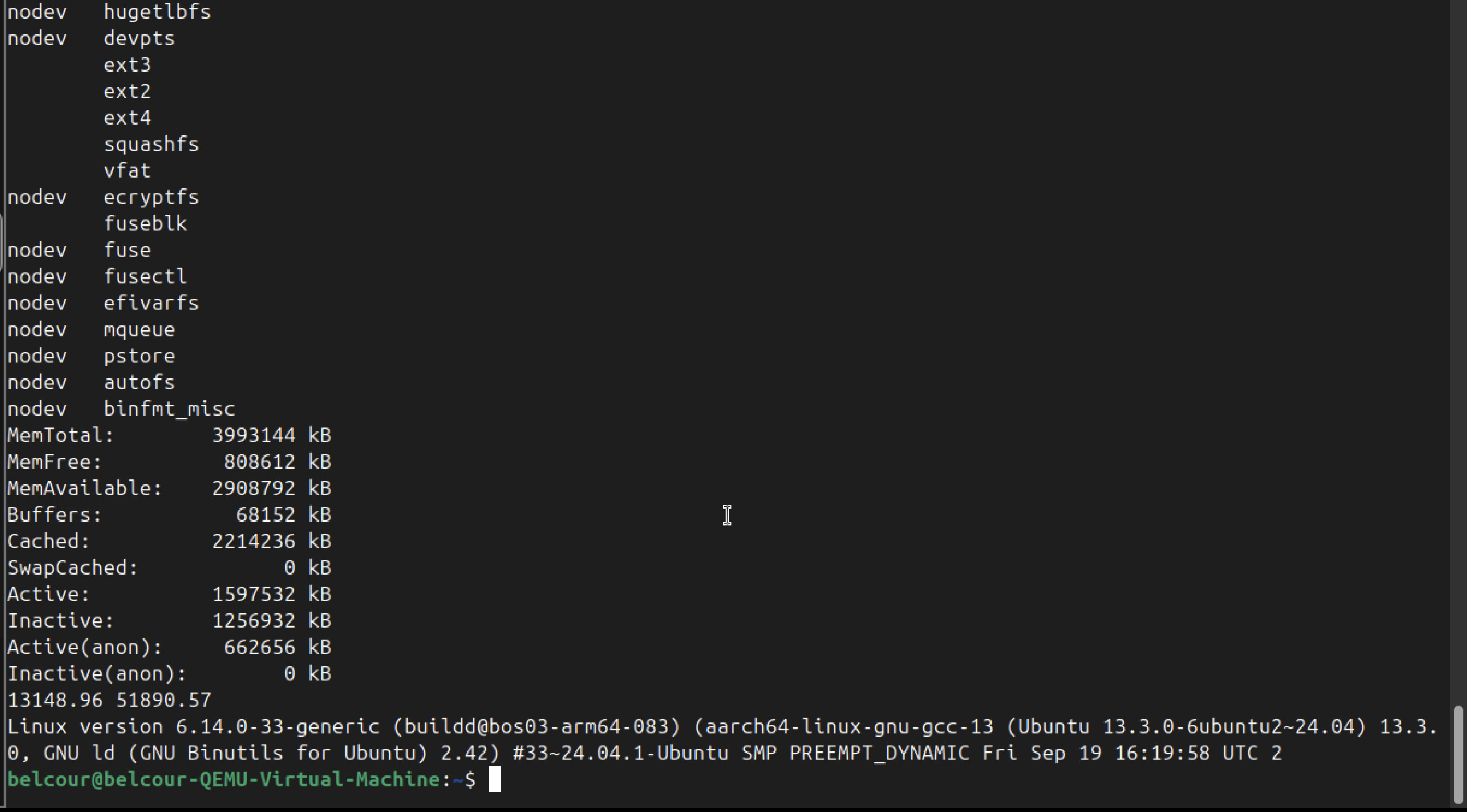


La commande mount montre que le système de fichiers proc est monté sur /proc.

Cela prouve que /proc est un pseudo-système géré directement par le noyau.

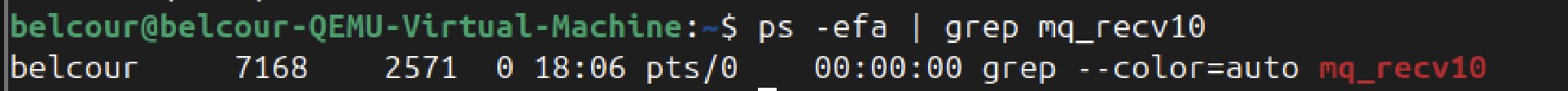
J’ai consulté ces fichiers à l’aide des commandes cat et head pour en afficher les premières lignes.

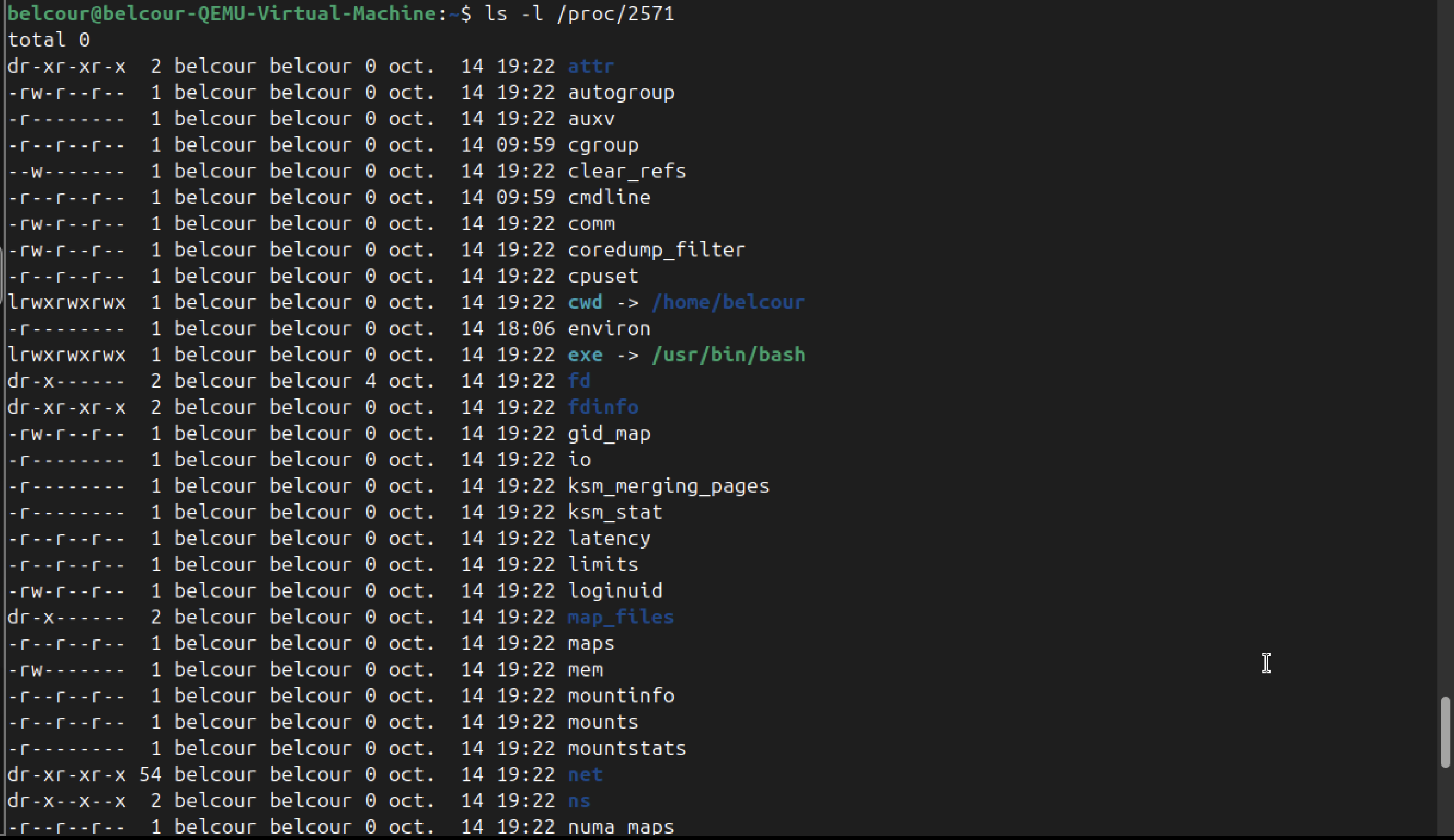


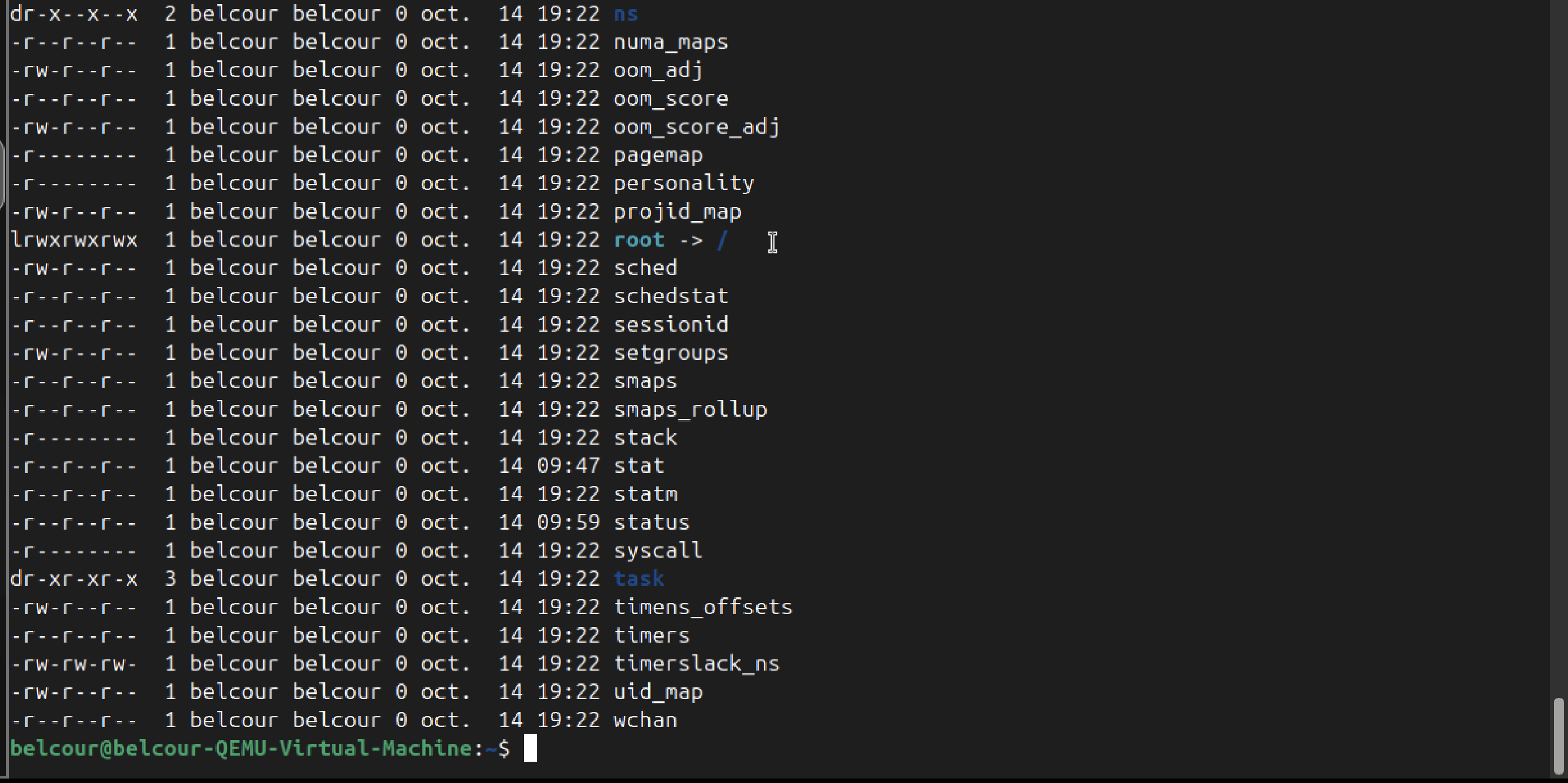


Ces fichiers m’ont permis de constater que les informations qu’ils contiennent (CPU, mémoire, modules, uptime, version du noyau) changent en temps réel et ne sont pas stockées sur le disque.

Pour analyser un processus précis, j’ai lancé mon programme mq\_recv10 puis recherché son PID avec :





Ce dossier contient plusieurs fichiers et liens symboliques :

**cmdline** → la commande exacte utilisée pour lancer le processus.

**cwd** → lien symbolique vers le répertoire courant du processus.

**environ** → variables d’environnement du processus.

**exe** → lien vers l’exécutable du processus.

**fd** → répertoire listant tous les descripteurs de fichiers ouverts par le processus.

**maps** → carte mémoire du processus.

Cela montre que /proc permet d’explorer en direct tous les détails internes d’un processus actif.

**cpuinfo** → infos détaillées sur le processeur (modèle, fréquence, nombre de cœurs, flags supportés…).

**devices** → liste des périphériques reconnus par le noyau.

**dma** → canaux DMA utilisés par les périphériques.

**filesystems** → types de systèmes de fichiers supportés par le noyau.

**interrupts** → table des interruptions, montre quelles IRQ sont utilisées par quels périphériques.

**partitions** → liste des partitions détectées par le noyau.

**meminfo** → état mémoire : RAM totale, libre, buffers, swap…

**modules** → liste des modules noyau actuellement chargés.

**uptime** → temps écoulé depuis le démarrage du système.

**version** → version du noyau Linux en cours d’exécution.