

Mini projet :

Il s'agit de saisir une valeur au clavier sur le PC pour afficher sa représentation décimale, hexadécimale et binaire sur un écran lcd connecté au port série du PC. L'heure locale du PC sera en même temps affichée périodiquement toutes les secondes sur l'écran lcd.

Objectifs :

- Programmation en langage C
- Configurer et piloter le port série d'un PC.
- Mise en œuvre d'une communication sur le port série, création de processus, de tubes de communication entre processus, gestion d'un segment de mémoire partagée, gestion d'un sémaphore et gestion de signal.
- Exploitation d'une documentation technique.



Le programme propose un menu qui permet d'écrire sur l'écran lcd une valeur saisie au clavier ou de quitter le programme sur l'entrée d'une valeur négative. On se limitera à saisir une valeur comprise entre 0 et 255 pour des raisons de simplicité et de capacité d'affichage de l'écran lcd.

```
Terminal
Fichier  Édition  Affichage  Rechercher  Terminal  Aide
etudiant@pent4 ~/rd/mini-projets/syst_ecran_clavier $
etudiant@pent4 ~/rd/mini-projets/syst_ecran_clavier $ ./appli
*****
* Affichage de l'heure (période 1s).                               *
* Conversion d'une valeur entiere (0 <= x < 256) en decimal, hexadecimal et binaire. *
*****
c - convertir une donnee.
q - sortir du programme.

Votre choix: c
Valeur a convertir: 113
c - convertir une donnee.
q - sortir du programme.

Votre choix: █
```

L'écran lcd affiche :

- Heure locale au format hh:mm:ss toutes les secondes, dès le démarrage du programme.
- Valeurs de l'octet saisi au clavier au format décimal, hexadécimal et binaire.



Figure 1: exemple d'affichage

L'affichage de l'heure est une tâche cyclique qui tourne en boucle de manière indépendante, on la dédiera donc à un processus fils. Le processus principal gérera les acquisitions clavier et l'écriture sur le port série.

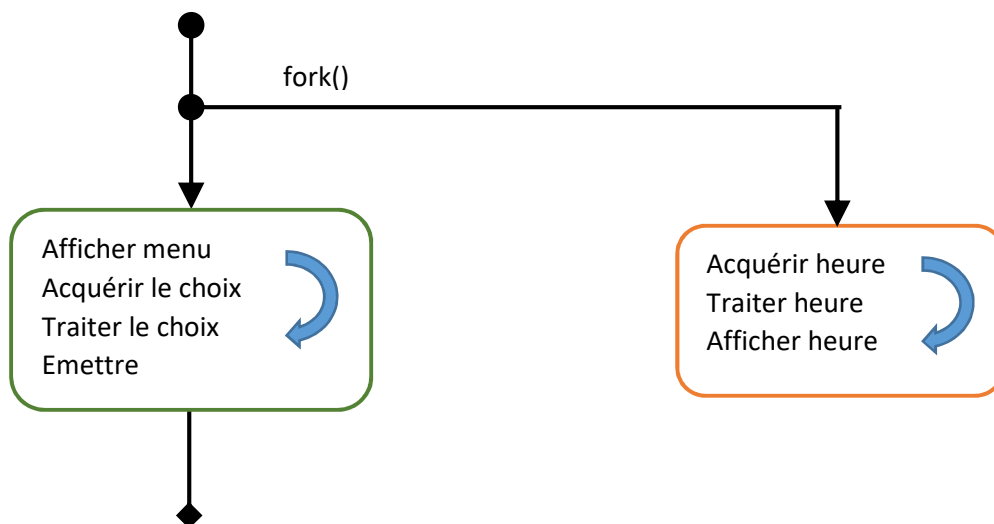


Figure 2: processus principal

processus fils

Les deux processus accédant à une ressource commune, l'écran lcd, une programmation correcte impose donc la mise en œuvre d'un sémaphore qui gérera l'accès à cette ressource pour éviter toute pollution de l'affichage.

Travail demandé :

Avant de démarrer :

- Tester la liaison série avec les commandes fournies en annexe 1.
- Tester la liaison en utilisant l'outil Putty.

L'écran est interfacé via une ligne série (/dev/ttyS0), on suppose connues les techniques d'accès à cette ligne. Les commandes permettant de gérer l'écran sont données dans le fichier lk204-35Ver2.0.pdf dans le répertoire miniProjet du serveur de fichiers.

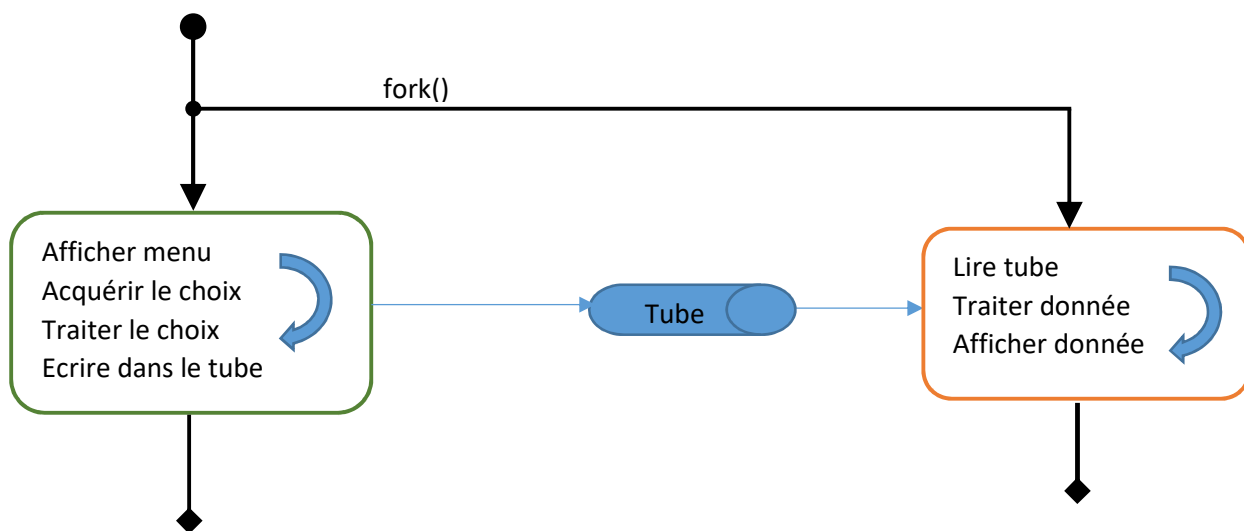
Pour démarrer, vous trouverez en annexe 2 un exemple de programmation de la ligne série sous Linux.

Point 1 : mettre au point le menu du processus principal. Faire l'acquisition d'une valeur numérique, la formater dans les trois bases voulues et les afficher sur les lignes 2, 3 et 4 de l'écran.

Point 2 : acquérir l'heure locale, la formater et l'afficher sur la ligne 1 de l'écran lcd.

Création d'un processus fils : faire tourner le travail 2 dans un processus fils (tâche rythmée de périodicité 1 seconde) et l'exécuter en parallèle avec le processus principal. Vérifiez par la commande `ps -ef` la présence des deux processus. Faites le choix de sortir dans le menu du processus principal et vérifiez que le processus fils est passé à l'état zombi et se trouve rattaché au processus n°1. Tuez le processus fils par la commande `kill -9 numero_de_pid`.

Mise en place d'un tube anonyme entre le processus père et le processus fils : pour terminer proprement le processus fils, mettez en place un tube anonyme entre les deux processus. Lors de la sortie du menu, le processus père écrira une valeur prédéfinie dans le tube. A réception, le processus fils sort de la tâche rythmée et se termine proprement par un appel à la fonction `exit`. Synchronisez les deux processus pour que le processus fils se termine correctement.



Processus de recouvrement : au niveau du fils, utilisez la fonction de recouvrement `execv()` pour mettre en place un exécutable qui jouera exactement le même rôle. Vérifiez que la synchronisation entre les deux processus est toujours effective.

Mise en place de tubes nommés.

Au démarrage de l'application, l'utilisateur doit s'authentifier via son nom et un code qui seront passés à un processus de contrôle d'accès **distinct** de nos deux premiers processus via un tube nommé. Ce processus aura été démarré avant votre application et tournera en tâche de fond. Le nom et le code sont entrés en ligne de commande lors du lancement de l'application.

Ex : `./a.out CROS 27212`. L'application doit d'abord vérifier qu'elle reçoit le bon nombre de paramètres sur la ligne de commandes et se termine sinon.

Nom et code sont transmis au processus de contrôle via le tube nommé qui consulte un fichier texte et vérifie si ils s'y trouvent. Il renvoie à l'application une autorisation de continuer ou non par un second tube nommé qui lui sert à répondre (cf. schéma situation finale).

Mise en place d'un segment de mémoire partagée.

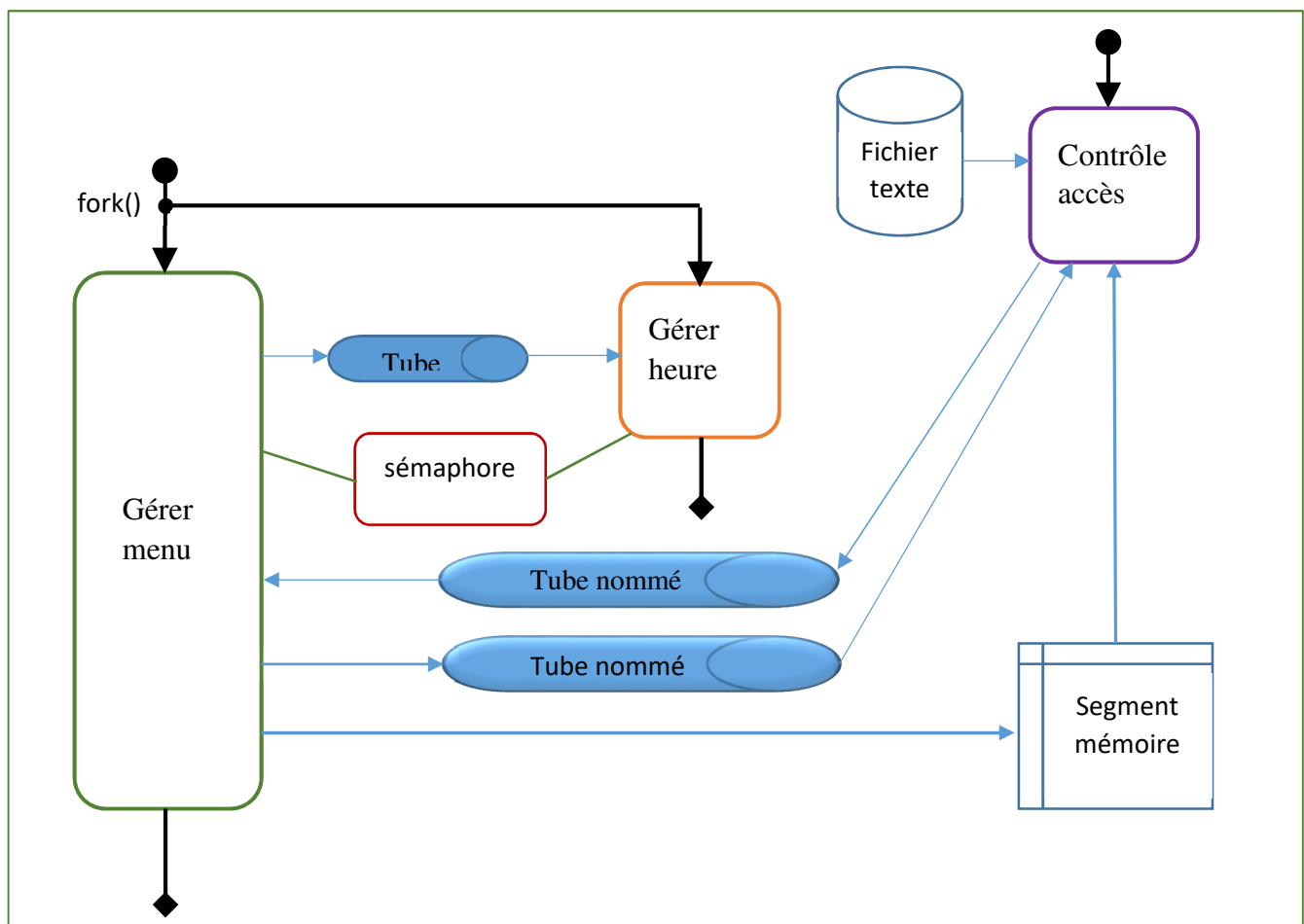
L'auteur de l'application veut connaître l'utilisation que vous en faites et mémoriser vos conversions pour une exploitation ultérieure (amélioration du produit, publicité ciblée, du big data donc...) Pour cela, on utilisera un segment de mémoire partagée qui servira à stocker le résultat des conversions. Lorsque l'application se termine, un ordre d'arrêt est passé au processus de contrôle d'accès précédent qui lira le contenu du segment de mémoire partagée pour l'afficher et le stocker dans un fichier texte conversions.data.

Implantation d'un mutex : pour mettre en évidence les problèmes qui peuvent survenir lors d'accès concurrents à une ressource commune, ralentissez de manière artificielle (sleep) le programme tournant dans le processus fils et constatez ce qui peut se passer à l'écran lors de la conversion d'une valeur. Pour cela il faudra disposer d'une fonction qui écrit dans le port série octet par octet.

Implantez un sémaphore de type mutex pour que les deux processus accèdent de manière sécurisée à l'écran lcd. Vérifier que le problème précédent disparaît.

Gestion de signal : implantez la gestion du signal SIGUSR1 pour mettre fin au processus fils. Visualisez le fait que le fils se termine brutalement en cours d'exécution.

Situation finale :



Annexe 1 : commandes et outils permettant de valider la liaison série de l'ordinateur en ligne de commande

ls -l /dev | grep ttyS0 → pour vérifier la présence du device ttyS0.

stty -a → pour afficher les paramètres de la liaison

sudo echo a > /dev/ttyS0 → envoie le caractère a sur /dev/ttyS0 via la liaison série.

Proposition d'algorithme de dépannage si l'affichage n'a pas lieu :

SI (le caractère a est affiché quelque part sur l'écran) {

 La liaison est valide.

 Sortir.

}

SINON {

// il se peut que vous ne puissiez pas accéder au port ttyS0.

 Exécutez la commande ls -l /dev | grep ttyS0 → le device ttyS0 appartient au groupe dialout.

 Exécutez la commande id pour afficher votre identifiant d'utilisateur.

 Exécutez la commande groups qui liste ceux auxquels vous appartenez.

 SI (vous appartenez au groupe dialout) {

 il y a un problème ailleurs.

 Sortir.

 }

 SINON {

 exécutez la commande sudo adduser *votre_identifiant* dialout

 Sortir.

 }

}

Redémarrez la machine.

Vérifiez que vous appartenez bien au groupe dialout.

Exécutez la commande : sudo echo a > /dev/ttyS0

Tester plus complètement la liaison avec l'outil Putty.

Annexe 2 : exemple d'initialisation de la ligne série. A simplifier et à adapter à vos besoins.

```
if ( (fd = open("/dev/ttyS0", O_RDWR)) == -1 ) {  
    perror("open");  
    exit(-1);  
}  
  
/* Lecture des paramètres courants de la liaison série */  
tcgetattr(fd, &termios_p);  
termios_p.c_cflag = B38400 | CS7 | CREAD;  
termios_p.c_lflag = ECHO;  
termios_p.c_oflag = OFDEL;  
termios_p.c_lflag = CSTOPB ;  
termios_p.c_cc[VMIN] = 1;  
termios_p.c_cc[VTIME] = 0;  
  
/* Sauvegarde des nouveaux paramètres de la liaison série */  
tcsetattr(fd, TCSANOW, &termios_p);
```

Pour plus d'informations, consultez la page : <https://www.cmrr.umn.edu/~strupp/serial.html> chapitre Setting Parity Checking