SNIR2 Programmation système – gestion de processus

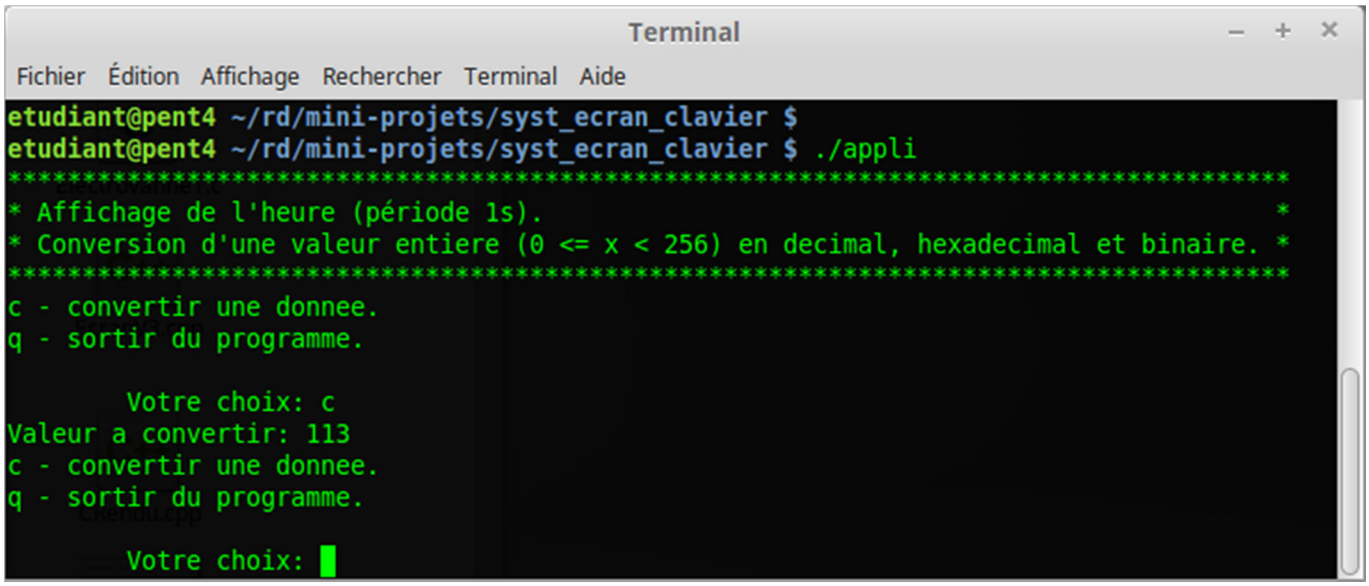
Mini projet :

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. L'heure locale du PC sera affichée périodiquement toutes les secondes sur l'écran lcd.

Objectifs :

- Configurer et piloter le port série d'un PC.
- Mise en œuvre d'une communication sur le port série, création de processus, de tube de communication entre processus 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.



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.

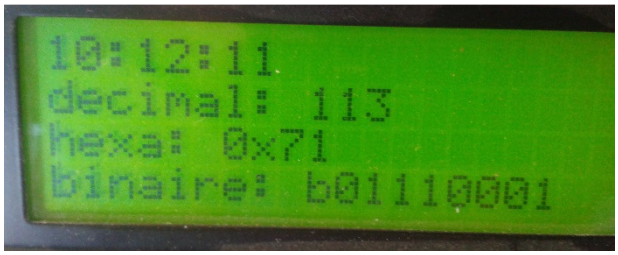


Figure 1: exemple d'affichage

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.

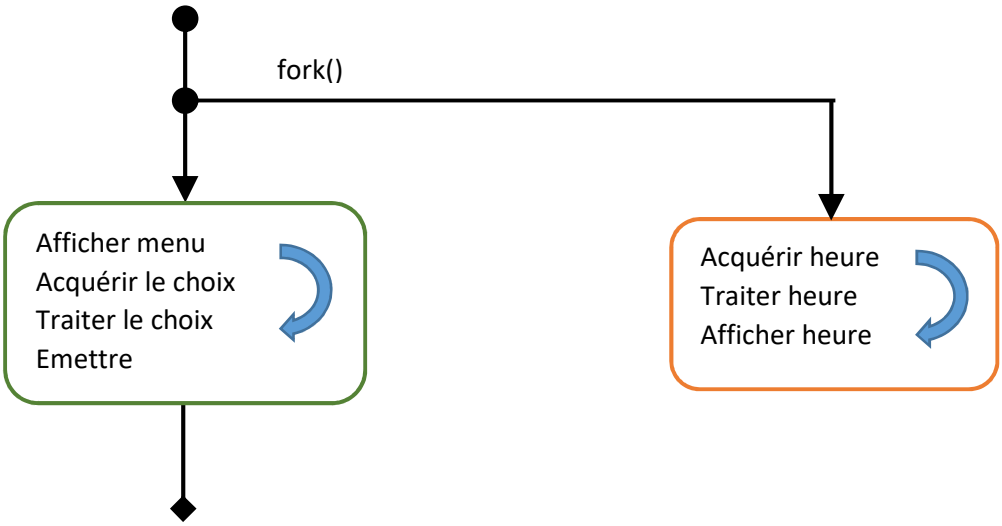


Figure 2: processus principal *processus fils*

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.

Travail 1 : mettre au point le menu du processus principal.

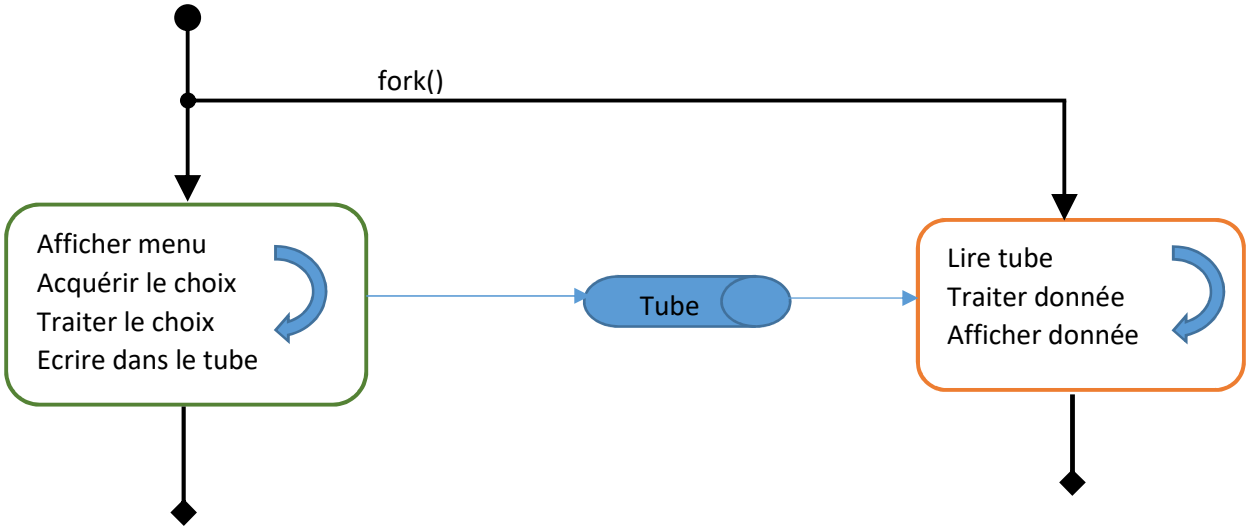
Travail 2 : acquérir l'heure locale, la formater et l'afficher sur la ligne 1 de l'écran lcd.

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.

Travail 3 : 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. Tentez de tuer le processus fils par la commande *kill -9 numero_de_pid*.

Travail 4 : pour terminer proprement le processus fils, mettez en place un tube anonyme entre les deux processus. Lors de la sortie du programme, le processus père écrira une valeur prédéfinie dans le tube. A réception, le processus fils sort de la tache rythmée et se termine par un appel à la fonction exit.



Si le processus fils affiche l'heure sans prendre en compte les secondes, la périodicité de la tache rythmée passe donc à 60 secondes. La technique utilisée précédemment convient-elle toujours pour une bonne expérience utilisateur ? Voir du côté de la fonction *select* pour s'affranchir du problème et modifiez le code en conséquence.

Travail 5 : pour mettre en évidence les problèmes qui peuvent survenir lors d'accès concurrents à une ressource commune, ralentissez de manière artificielle 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.

Travail 6 : synchroniser les deux processus pour que le processus fils se termine correctement.

Travail 7 : 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.

Travail 8 : pour terminer, 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.

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

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 → envoi le caractère a sur /dev/ttyS0 via la liaison série.

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 */
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 */
tcsetattr(fd, TCSANOW, &termios_p);
```