INSA
INSTITUT NATIONAL
DES SCIENCES
APPLIQUÉES
CENTRE VAL DE LOIRE

Module : Système II
Entrées Sorties avancées et Sockets domaine Unix

Promo 2020

STI 3A

2017_2018

But du cours

- Permettre d’optimiser le fonctionnement des entrées-sorties (sur périphériques, tubes, sockets) pour un thread ou un processus.
- Multiplexer les échanges entrées-sorties en les rendant asynchrones.

L’attribut O_NONBLOCK et l’appel système « open »

- Lors de l’ouverture d’un descripteur :
  - L’appel system « open » permet, par l’intermédiaire de l’attribut O_NONBLOCK, de rendre l’écriture ou la lecture non bloquante.
  - Ceci se règle lors de la création du descripteur.
  - Toute lecture sur ce descripteur sera non bloquante si aucune donnée n’est arrivée.
  - Toute écriture sur ce descripteur sera non bloquante si par exemple le buffer est plein.
  - Inconvénient : ceci n’est valable que si le descripteur est obtenu par « open » et non par (mkfifo, popen, socket..)

L’attribut O_NONBLOCK sans l’appel système « open »

- Si la fonction utilisée pour obtenir le descripteur n’est pas « open() » alors on utilise la fonction fcntl() pour rendre le descripteur non bloquant.

  - prototype : fcntl (int fd, int commande, …);
    exemple :
    fcntl (fd1, F_SETFL, fcntl (fd, F_GETFL) | O_NONBLOCK);

Attente de lectures ou de possibilités d'écritures non bloquantes

- Variable nbdescripteur type entier
- Variable i type entier
- Initialisation nbdescripteur exemple 10
- Tant que 1 vrai # début boucle infinie
  - Pour i ← 1 à nbdescripteur
    - Si lecture sur descripteur n° i possible
      - Alors traiter les données reçues
    - Fin si
  - Fin pour
- Fin tant que

- Inconvénient utilisation de temps CPU pour tourner en boucle souvent sans traitement ou mettre au repos sleep().

Multiplexage d'entrée : select()

- Des applications ont parfois besoin de surveiller l'arrivée de données venant de descripteurs multiples et réveiller le processus ou le thread.
- Le noyau étant le premier à gérer ces événements, il lui est possible de prévenir l'application si des données sont disponibles sur un descripteur parmi un ensemble de descripteurs.
- L'appel système « select() » permet de gérer ce cas de figure.
- Prototypage de select :
  - int select (int numgranddescripteur+1, fd_set *ensemblealire, fd_set *ensembleaecrire, fd_set *ensembleexeptions, struct timeval * delai_maxi);

Les paramètres de select()

- int select (int numgranddescripteur+1, fd_set *ensemblealire, fd_set *ensembleaecrire, fd_set *ensembleexeptions, struct timeval * delai_maxi);
- int numgranddescripteur : numéro du plus grand descripteur à surveiller+1 (voir exemple exemple_select.c).
- fd_set *ensemblealire pointe sur un type opaque contenant la liste des descripteurs à surveiller en lecture.
- fd_set *ensembleaecrire idem pour l'écriture.
- fd_set *ensembleexeptions descripteur sur l'arrivée de conditions exceptionnelles (EX : données hors bande).
- Permet de réveiller un processus après un délai d'attente maxi.

Entrées sorties asynchrones et fcntl()

- En utilisant la possibilité de pouvoir utiliser un descripteur lors de sa disponibilité et de pouvoir continuer à exécuter du code.
  Principe :
  - On associe le pid du processus au descripteur de fichier.
    fctnl (fd, F_SETOWN, getpid());
    on indique le pid du processus devant recevoir le signal
  - On laisse le noyau, via un signal, nous prévenir de la modification du descripteur.
    fcntl (fd, F_SETSIG,numerosignal);
    Si numerosignal==0 alors SIGIO est employé.
    L'utilisation d'un signal temps réel permet d'obtenir des informations supplémentaires dans une structure siginfo.
    En particulier :
    le membre si_code contient la valeur SI_SIGIO
    le membre si_fd contient la valeur du descripteur disponible.
  - On met en place un gestionnaire de signal temps réel.
- Voir exemple_async.c

Entrées sorties asynchrones POSIX 1.B

- Inconvénient de select et de fcntl est de détecter le changement d'état d'un descripteur mais pas la quantité de données disponible. Le déclenchement peut se faire pour 1 octet.
- Solution :
  - Programmer l'opération de lecture ou d'écriture.
  - Gestion par le noyau de l'opération.
  - Le noyau prévient l'application de la fin de l'opération.
- Mise en œuvre :
  On prépare un bloc sous la forme d'une structure aiocb qui contient entre autre :
  - Un buffer
  - Le descripteur de fichier.
  - Le type d'opération
  - …

Entrées sorties asynchrones POSIX 1.B

- Contenu de la structure « aiocb »

aio_fildes	int	Descripteur concerné
aio_offset	off_t	Offset dans le fichier/début
aio_buf	void *	Adresse du buffer
aio_nbytes	size_t	Nombres d'octets à traiter
aio_reqprio	int	Niveau de priorité
aio_sigevent	struct sigevent	Mécanisme de signalisation
aio_lio_opcode	int	Code opératoire

Entrées sorties asynchrones POSIX 1.B

- Le membre aio_reqprio indique une valeur soustraite à la priorité du processus. Si on augmente cette valeur la priorité chute.
- Le membre aio_sigevent permet d'envoyer un signal ou de démarrer un thread sur une fonction choisie. struct sigevent :

Sigev_notify	Int soit SIGEV_NONE, SIGEV_SIGNAL, SIGEV_THREAD	Type notification
Sigev_signo	Int	Numéro signal
Sigev_value	Sigval_t	Valeur pour gestionnaire, thread
Sigev_notify_function	Void(*f) (sigval_t), · Union sigval	Fonction à déclencher thread
Sigev_notify_attributes	Pthread_attr_t	Attribut du thread

Entrées sorties asynchrones POSIX 1.B

- Le membre aio_lio_opcode de la structure aiocb est peu utilisé en effet les opérations de lecture et d'écriture utilisent les fonctions :
  - Lecture :
    int aio_read (struct aiocb *aiocb);
  - Ecriture :
    int aio_write (struct aiocb *aiocb);
- Voir le programme exemple_aio_read.c

Socket du domaine Unix

- Les sockets du domaine Unix sont couramment utilisées pour des processus s'exécutant sur le même calculateur.
- Un exemple classique est le démon syslogd, voir la socket locale « /dev/log ». En écrivant sur cette socket toute application peut transmettre des données à syslogd. Voir la commande logger et le résultat dans /var/log/messages.
- Ces sockets se manipulent comme des sockets de type internet AF_INET ou PF_INET sauf pour l'adressage.
- Adressage AF_UNIX ou PF_UNIX
  - struct sockaddr_un {
  - short sun_family; /* AF_UNIX */
  - char sun_path[108]; /* path name */
  - };
- Voir les exemples sur le serveur enseignement.insa-cvl.fr