Facultat d'Informàtica de Barcelona
UNIVERSITAT POLITÈCNICA DE CATALUNYA

CONCEPTES AVANÇATS DE SISTEMES OPERATIUS
Departament d'Arquitectura de Computadors

Native POSIX Thread Library for Linux

(Seminaris de CASO)

Autors

Juan Belmonte Rodríguez

Roger Ferrer Ibáñez

Grup T15

Què és Native Posix Thread Library (NPTL) ?

- És una nova implementació de POSIX Threads a Linux.
  - Ja està disponible en les distribucions més recents de GNU/Linux com ara Red Hat 9.
- La implementació que s'ha vingut usant fins ara s'anomena LinuxThreads.
- NPTL proposa una implementació nova que millora i corregeix algunes deficiències de la implementació actual.
  - Eficiència, POSIX-compliance issues, etc.

Models de threading

- En un model 1-M, l'usuari s'encarrega d'emular els threads mitjançant llibreries: 1 procés de nucli - M "threads" emulats.
  - Aquesta solució és ineficient i no permet aprofitar els sistemes multiprocessador.
- En un model 1-1, cada thread d'usuari s'assigna a una "unitat de planificació" (anomenats kernel-threads) del nucli.
- En un model M-N, s'usen dos planificadors. Un planificador en espai d'usuari que "aconsella" al planificador del nucli. Aquest últim notifica al planificador d'usuari les seves decisions de planificació. (Scheduler Activations).
  - Implementació usada a Solaris o a NGPT (Next Generation POSIX Threads)

Relació entre el kernel Linux i LinuxThreads

- Algunes característiques del kernel Linux van condicionar la implementació LinuxThreads.
- Una única crida al sistema clone(2) que crea un kernel thread
  - Això va dur a un model d'implementació 1-1
- No hi ha primitives de sincronització *útils* a espai d'usuari.
  - Això va dur a una implementació de la sincronització dels threads basada en signals i l'existència d'un procés gestor dels threads.

Inconvenients de LinuxThreads

- Un thread = un procés de kernel. Per cada thread que es crea apareix una entrada al directori /proc.
  - Cada thread té el seu propi PID ! (Contràriament a POSIX)
  - L'arquitectura IA-32 només pot tenir fins a 8192 threads.
- La sincronització amb signals és fràgil i afegeix molta latència al sistema.
  - Alguns signals enviats al procés que conté els threads s'han de tractar de forma especial (p.e. SIGSTOP i SIGCONT)
- El procés gestor de threads és un coll d'ampolla.
  - A més, si el procés gestor rep un SIGKILL es perd tota la funcionalitat dels threads.

Evidència del thread gestor

- Aquest és el resultat d'executar una aplicació que crea un dos threads. Hem fet que es bloquegin per tal de veure els processos.

```
[roger@caecus ~/CASO/seminari]$ ./thread_gest &
[1] 32590
Soc el pare dels fills i tinc pid 32590
Soc el thread 0 i tinc pid 32592
Soc el thread 1 i tinc pid 32593
[roger@caecus ~/CASO/seminari]$ ps -U roger f
  PID TTY      STAT   TIME COMMAND
32083 ?        SN     0:03 /usr/sbin/sshd
32084 pts/0    SN     0:00  \_ -bash
32590 pts/0    SN     0:00      \_ ./thread_gest
32591 pts/0    SN     0:00      |   \_ ./thread_gest
32592 pts/0    SN     0:00      |       \_ ./thread_gest
32593 pts/0    SN     0:00      |       \_ ./thread_gest
32594 pts/0    RN     0:00      \_ ps -U roger f
```

La nova implementació (1)

- Els canvis s'han fet tant a nivell de kernel com a nivell de llibreria del sistema (GNU Libc).
- Recentment s'ha proposat una nova primitiva de sincronització d'espai d'usuari anomenada *futex*.
  - És molt més ràpida que d'altres primitives com les de System V (que requereixen una crida al sistema cada cop).
  - No cal usar signals per sincronitzar els processos : menor latència.

La nova implementació (i 2)

- Segueix usant un model 1-1
  - Els desenvolupadors del kernel consideren que M-N no es pot implementar actualment i consideren que les rutines de planificació del kernel són suficientment eficients.
- No hi ha un thread gestor.
- El PID dels threads és el del procés que els conté.
  - No s'emplena el directori /proc per cada thread creat.
- El kernel suporta un número arbitrari de threads.
  - L'arquitectura IA32 suporta 2 bilions de threads.

Exemple : Test de rendiment

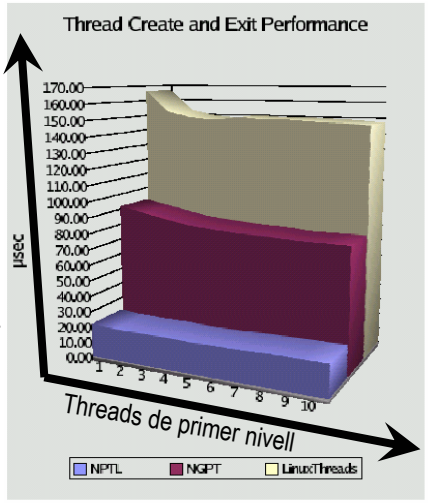

Figure 2: Varying number of Concurrent Children

Aquest diagrama s'ha extret de la referència [1]

Consisteix en comptabilitzar el temps que costa crear un thread en el següent codi.

for 1 to N threads de primer nivell
crear per cada thread de primer nivell
10 threads fills.

L'exemple anterior en NPTL

- Aquest és el mateix programa d'abans per veure el thread gestor. En aquest cas s'ha executat sota un Red Hat 9 amb NPTL. No hi ha tread gestor i els PID's són els correctes.

```
[roger@pandora-rh roger]$ ./thread_gest &
[1] 1571
Soc el pare dels fills i tinc pid 1571
Soc el thread 0 i tinc pid 1571
Soc el thread 1 i tinc pid 1571
[roger@pandora-rh roger]$ ps -U roger f
  PID TTY      STAT   TIME COMMAND
 1525 ?        S      0:00 [sshd]
 1526 pts/0    S      0:00 -bash
 1571 pts/0    T      0:00  \_ ./thread_gest
 1575 pts/0    R      0:00  \_ ps -U roger f
```

Codi de l'exemple (1)

```
pthread_mutex_t mux_printf;
int num_thread = 0;

void* codi_fil(void* data) {
       sem_t bloqueig;

       pthread_mutex_lock(&mux_printf);
       printf("Soc el thread %d i tinc pid %d\n",
                     num_thread, getpid());

       num_thread++;
       pthread_mutex_unlock(&mux_printf);

       // Usem un semafor per provocar el bloqueig
       sem_init(&bloqueig, 0, 0);
       sem_wait(&bloqueig);
}
```


Codi de l'exemple (i 2)

```
int main (int argc, char* argv[]) {
       int i;
       sem_t bloqueig;
       pthread_t fil;

       printf("Soc el pare dels fills i tinc pid %d\n",
              getpid());
       pthread_mutex_init(&mux_printf, NULL);
       for (i = 0; i < 2; i++) {
              pthread_create(&fil, NULL, codi_fil, NULL);
       }

       // Usem un semafor per provocar el bloqueig
       sem_init(&bloqueig, 0, 0);
       sem_wait(&bloqueig);
}
```

Bibliografia

- [1] U. Drepper, I. Molnar. *The Native POSIX Thread Library for Linux*. (Draft) Gener 2003.
  - http://people.redhat.com/drepper/nptl-design.pdf
- [2] H. Franke, R. Russell, M. Kirkwood. *Fast Userlevel Locking in Linux*. Proceedings of the Ottawa Linux Symposium 2002.
  - http://www.linux.org.uk/~ajh/ols2002_proceedings.pdf.gz
- [3] U. Drepper et al. Design of the New GNU Thread Library.
  - http://people.redhat.com/drepper/glibcthreads.html

Referències

- Pàgina de LinuxThreads
  - http://pauillac.inria.fr/~xleroy/linuxthreads/
- Pàgina de Next Generation POSIX Threads
  - http://www-124.ibm.com/developerworks/oss/pthreads/