Threads

Veerle Ongenae

Threads

- Inleiding
- Threads maken
- Argumenten meegeven aan threads
- Wachten op einde van een thread
- Thread return waarden
- Thread beëindigen
- Voorbeelden

Inleiding

- Lichtgewicht proces
- Threads lopen schijnbaar tegelijk
- Bestaan binnen een proces
- Threads delen globale variabelen
- Bestandsdescriptoren worden ook gedeeld

Inleiding

- Netwerkprogrammatie
  - Voor elke client een aparte thread
  - Aparte thread lezen invoerkanaal
- POSIX threads best ondersteund
- Compileren met -lpthread
  - Gebruik bibliotheek (library) pthread

Threads maken

int **pthread_create**(pthread_t *thread,
const pthread_attr_t attr,
void *(*func)(void *),
void *arg);

return O als ok, niet-nul bij fout

- maakt nieuwe thread met eigenschappen uit attr (doorgaans NULL)
- Start functie func() en geeft argument arg door.

pthread_create

- thread
  - Pointer naar id van de nieuwe thread
- attr
  - Hoe interactie met rest programma
  - Null
    - Standaardwaarden
- func: pointer naar functie
- arg: data voor thread

Threads maken

- pthread_create keert onmiddellijk terug
- De volgorde van uitvoering van de code in de threads is onvoorspelbaar

Einde threads

- thread stopt
  - threadfunctie terugkeert
  - binnen de thread de functie pthread_exit aanroepen
- in een thread de functie exit aanroepen
  - alle threads binnen het proces
- hoofdthread stopt (main keert terug)
  - alle threads binnen het proces stoppen

Voorbeeld

- twee_oneindige_threads.cpp
- Opstarten thread
  - 1'en schrijven
- Hoofdthread
  - 0'en schrijven

Argumenten doorgeven

- Globale variabelen :
  - Eenvoudig
  - Gevaarlijk want twee threads kunnen tegelijk naar 1 variabele schrijven
- Beter via laatste parameter van pthread_create
  - Slechts 1 argument van type void * mogelijk --> casten
  - Indien meer argumenten moeten worden doorgegeven : gebruik array (tabel) of struct

Voorbeeld

- twee_threads_schrijven.cpp
- Twee threads op starten
- Elke thread schrijft een bepaald karakter een aantal keer uit
- Karakter + aantal keer
  - struct
- Probleem ...

Wachten op einde thread

int **pthread_join**(
  pthread_t thread,
  void **thread_return)

return : 0 bij ok, positief bij fout

- thread is de id van de thread waarop je wil wachten tot hij afloopt
- thread_return bevat de return waarde van de thread wanneer hij afloopt

Voorbeeld aangepast

- twee_threads_schrijven_v2.cpp

Return waarden

- Kleine waarden via parameter pthread_join
  - Werkt enkel voor <= 4 bytes want pointer_variabele (of 8 bytes afhankelijk van systeem)
  - void *
  - Eigenlijk voor exitstatus
- Voor grotere of complexere uitvoer : gebruik de parameter die je meegeeft aan pthread_create()

Voorbeeld

- kwadraat_in_thread.cpp
- kwadraat_in_thread_v2.cpp
  - Overschrijven oorspronkelijke variabele
- grootste_in_tabel.cpp
- grootste_in_tabel_double.cpp

Threads beëindigen

- Return threadfunctie
- Return main
- Oproepen exit in (andere) thread
- Oproepen pthread_exit
  - Zichzelf beëindigen
- Oproepen pthread_cancel
  - Annuleren

pthread_exit

#include <pthread.h>

void **pthread_exit** (void *status);

- Status: return waarde thread
  - Kan opgevraagd via pthread_join

Threads annuleren

- Je kan een thread beëindigen met de functie pthread_cancel().

int **pthread_cancel**(pthread_t thread)

return 0 als ok, positief bij error

- gecancelde thread moet later met de functie pthread_join opgewacht worden opdat de bronnen worden vrijgegeven.

Threads annuleren

- return waarde van een gecancelde thread (op te vragen in pthread_join) is PTHREAD_CANCELED
- thread kan zelf bepalen wanneer hij kan geannuleerd worden :
  - asynchroon : op elke punt te annuleren
  - synchroon : enkel op bepaalde punten
    - bv. oproep read, write, ...
  - uncancelable : nooit

Threads annuleren

- het instellen van deze toestanden gebeurt met de functie

int **pthread_setcanceltype**(
  int type,
  int *old_type)

return 0 als ok, positief bij error

- vorige type wordt opgeslagen in old_type

Mogelijke instellingen

- PTHREAD_CANCEL_ASYNCHRONOUS
  - Asynchroon
- PTHREAD_CANCEL_DEFERRED
  - Synchroon
  - Standaard cancel-toestand

Cancel points

- cancel points kan je zelf toevoegen met de functie pthread_testcancel.
  - Is er een annulatie-aanvraag?
    - Ja: stoppen
    - Neen: verder doen

Uncancelable Thread

- een thread maak je uncancelable met de functie pthread_setcancelstate :

int **pthread_setcancelstate**(
  int state,
  int *old_state)
return 0 indien OK, positief indien error

Uncancelable Thread

- state
  - PTHREAD_CANCEL_ENABLE
  - PTHREAD_CANCEL_DISABLE

Voorbeeld

```
int transactie(float rekeningen[ ], int van_rekening, int naar_rekening,
    int bedrag){
  int old_state;
  if (rekeningen[van_rekening] < bedrag)
    return 1; // fout ! niet genoeg geld op rekening

  // begin kritische sectie
  pthread_setcancelstate          PTHREAD_CANCEL_DISABLE,
    &old_state);
  rekeningen[van_rekening] -= bedrag;
  rekeningen[naar_rekening] += bedrag;
  // einde kritische sectie
  pthread_setcancelstate(old_state, NULL);

  return 0;
}
```

Voorbeeld: EchoServer

- echo_server.cpp