Sleeping_TA

AUTHOR: Dr. Christer Karlsson, Aaron Alphonsus
Version: 1.0
DATE: 27 February 2017

File Index

File List

Here is a list of all files with brief descriptions:

File Documentation

hangout.c File Reference

Simulate a student hanging out.
`#include <stdio.h>`

Functions

- void **hang_out** (int lnumber, int sleep_time)

Detailed Description

Simulate a student hanging out.

Authors:
Dr. Christer Karlsson, Aaron Alphonsus

Date:
27 February 2017

Function Documentation

void hang_out (int *lnumber*, int *sleep_time*)

Function takes in a studentid and time in seconds. Prints the student id with the time it is hanging out and sleeps for that time.

Parameters:

in	*lnumber*	ID of the student
in	*sleep_time*	Number of seconds to sleep for

Prints the id of the student and time that it hangs out

Sleep for given number of seconds

help_student.c File Reference

Simulate helping a student.

```
#include <stdio.h>
#include "ta.h"
```

Functions

- void **help_student** (int sleep_time)

Detailed Description

Simulate helping a student.

Authors:

Dr. Christer Karlsson, Aaron Alphonsus

Date:

27 February 2017

Function Documentation

void help_student (int *sleep_time*)

Function takes in a time in seconds. Prints the time that the student is helped with the number of waiting students, and sleeps for the given time.

Parameters:

in	*sleep_time*	Number of seconds to sleep for

Print time that student is helped and number of waiting students

Sleep for given number of seconds

main.c File Reference

Main file for sleeping TA. Program execution begins by calling the **init()** function which initializes the student_id array, mutex and semaphores. It then creates a TA thread to run **ta_loop()**, and student threads to run **student_loop()**. Program execution in concluded by joining the student threads and canceling the TA thread.

```
#include <pthread.h>
#include <stdio.h>
#include <semaphore.h>
#include <string.h>
#include <errno.h>
#include "ta.h"
```

Functions

- void **init** ()
- void **create_students** ()
- void **create_ta** ()
- int **main** (void)

Variables

- pthread_t **ta**
  Define TA thread.
- pthread_t **students** [**NUM_OF_STUDENTS**]
  Define student threads.

Detailed Description

Main file for sleeping TA. Program execution begins by calling the **init()** function which initializes the student_id array, mutex and semaphores. It then creates a TA thread to run **ta_loop()**, and student threads to run **student_loop()**. Program execution in concluded by joining the student threads and canceling the TA thread.

Authors:

Dr. Christer Karlsson, Aaron Alphonsus

Date:

27 February 2017

Function Documentation

void create_students ()

This function creates a thread for each student to execute **student_loop()** and pass in its student id as a parameter.

Loop through number of students creating student threads

void create_ta ()

This function creates a TA thread to execute **ta_loop()**

Create TA thread to run **ta_loop()**

void init ()

This function initializes all relevant variables, data structures and synchronization objects.

Initialize waiting students to 0

Initialize student being served to -1

Initialize mutex

Initialize student semaphore

Initialize TA semaphore

Initialize student id array

int main (void)

This is the main function. It makes function calls to initialize variables and synchronization objects, create the TA thread, and create student threads. It then joins the student threads, followed by the cancellation of the TA thread.

Returns:

0 Indicates normal termination of main.

Declare iterator variable

Function call to initialize variables and synchronization objects

Function call to create TA thread to execute **ta_loop()**

Function call to create student threads to execute **student_loop()**

Join student threads

When all students have finished, we will cancel the TA thread

---

Variable Documentation

pthread_t students[NUM_OF_STUDENTS]

pthread_t ta

Define TA and student threads.

student.c File Reference

General structure of a student.

```
#include <pthread.h>
#include <stdio.h>
#include <time.h>
#include <errno.h>
#include <string.h>
#include "ta.h"
```

Functions

- void * **student_loop** (void *param)
  Student function prototype.

Detailed Description

General structure of a student.

Authors:

Dr. Christer Karlsson, Aaron Alphonsus

Date:

27 February 2017

Function Documentation

void* student_loop (void * *param*)

Student function prototype.

This function simulates student behavior. We first acquire the mutex lock. If there is an available seat, we increment the number of waiting students, print the student which takes a seat and the number of students waiting. We then release the mutex, and use the student and TA semaphores to track the students waiting on the TA and whether the TA is free. If there is no available seat, we unlock the mutex and call the hangout function so that the student can try again later.

Parameters:

in	*param*	Pointer to an int pointer to the student id

Declare local variables

Seed random generator

Student leaves after going for help 5 times

Acquire the mutex lock

Check for available seat, else hangout.

Increment number of waiting students

Print which student takes a seat and number waiting.

Release mutex lock
Signals students waiting for TA
Wait until TA isn't helping another student
Print which student is receiving help
No available seat
Print student will try later
Release mutex lock
Function call simulating student hanging out
Increment loop counter

ta.c File Reference

General structure of the teaching assistant.

```
#include <pthread.h>
#include <stdio.h>
#include <time.h>
#include <errno.h>
#include <string.h>
#include "ta.h"
```

Functions

- void * **ta_loop** (void *param)
  TA function prototype.

Detailed Description

General structure of the teaching assistant.

Authors:

Dr. Christer Karlsson, Aaron Alphonsus

Date:

27 February 2017

Function Documentation

void* ta_loop (void * *param*)

TA function prototype.

This function defines the behavior of the TA. We perform a sem_wait to wait for a student to show up. When a student shows up, we acquire the mutex, post the TA semaphore, and release the mutex. After this, we make a call to **help_student()** to simulate helping a student for a random amount of time.

Parameters:

in	*param*	Void pointer

Seed random generator

Wait for a student to show up

Acquire the mutex lock

Decrement number of waiting students

Indicate the TA is ready to help a student

Release mutex lock

Help students random time

ta.h File Reference

Header file for sleeping TA. Contains definitions for a number of constants, as well as function prototypes and variable declarations (includes declarations of the mutex lock and semaphores.)

```
#include <pthread.h>
#include <semaphore.h>
```

Macros

- #define **MAX_SLEEP_TIME** 5
  The maximum time (in seconds) to sleep.
- #define **MAX_WAITING_STUDENTS** 3
  Number of maximum waiting students.
- #define **NUM_OF_STUDENTS** 5
  Number of potential students.
- #define **NUM_OF_SEATS** 3
  Number of available seats.

Functions

- void * **student_loop** (void *param)
  Student function prototype.
- void * **ta_loop** (void *param)
  TA function prototype.

Variables

- pthread_mutex_t **mutex_lock**
  Mutex lock.
- sem_t **student_sem**
  Semaphore declarations.
- sem_t **ta_sem**
- int **waiting_students**
  The number of waiting students.
- int **student_served**
  Student being served.
- int **student_id** [**NUM_OF_STUDENTS**+1]
  The numeric id of each student.

Detailed Description

Header file for sleeping TA. Contains definitions for a number of constants, as well as function prototypes and variable declarations (includes declarations of the mutex lock and semaphores.)

Authors:

Dr. Christer Karlsson, Aaron Alphonsus

Date:
27 February 2017

Macro Definition Documentation

#define MAX_SLEEP_TIME 5

The maximum time (in seconds) to sleep.

#define MAX_WAITING_STUDENTS 3

Number of maximum waiting students.

#define NUM_OF_SEATS 3

Number of available seats.

#define NUM_OF_STUDENTS 5

Number of potential students.

Function Documentation

void* student_loop (void * *param*)

Student function prototype.

This function simulates student behavior. We first acquire the mutex lock. If there is an available seat, we increment the number of waiting students, print the student which takes a seat and the number of students waiting. We then release the mutex, and use the student and TA semaphores to track the students waiting on the TA and whether the TA is free. If there is no available seat, we unlock the mutex and call the hangout function so that the student can try again later.

Parameters:

in	*param*	Pointer to an int pointer to the student id

Declare local variables

Seed random generator

Student leaves after going for help 5 times

Acquire the mutex lock

Check for available seat, else hangout.

Increment number of waiting students

Print which student takes a seat and number waiting.

Release mutex lock

Signals students waiting for TA

Wait until TA isn't helping another student

Print which student is receiving help

No available seat

Print student will try later

Release mutex lock

Function call simulating student hanging out

Increment loop counter

void* ta_loop (void * *param*)

TA function prototype.

This function defines the behavior of the TA. We perform a sem_wait to wait for a student to show up. When a student shows up, we acquire the mutex, post the TA semaphore, and release the mutex. After this, we make a call to **help_student()** to simulate helping a student for a random amount of time.

Parameters:

in	*param*	Void pointer

Seed random generator

Wait for a student to show up

Acquire the mutex lock

Decrement number of waiting students

Indicate the TA is ready to help a student

Release mutex lock

Help students random time

Variable Documentation

pthread_mutex_t mutex_lock

Mutex lock.

int student_id[NUM_OF_STUDENTS+1]

The numeric id of each student.

sem_t student_sem

Student semaphore

int student_served

Student being served.

sem_t ta_sem

TA semaphore

int waiting_students

The number of waiting students.