CS518 Design of Operating System

Fall 2022

Report

User-level thread library

iLab machine tested on: -ilab1.cs.rutgers.edu

VISHWAS GOWDIHALLI MAHALINGAPPA
vg421

VIKRAM SAHAI SAXENA
vs799

1 Introduction

Develop a user-level thread library utilizing the pthread library as a model and using the ucontext library to create and swap threads, and the return pointer is set to a default context that only does a mypthread exit (). After creating the threads, scheduler then executes a thread by assigning it to the scheduling algorithms (RR, PSJF and MLFQ) and it uses mutexes to implement mutual exclusion using test and set atomic operation. The scheduler runs every so often using setitimer() to cause a single signal at a set duration or to periodically raise a signal at regular intervals and signal() to catch it in order to run the scheduler code.

2 Design and Implementation

Structs and Enums:

- *Thread status enum:*
    - *Contains states of threads in thread control block.*

- *threadControlBlock struct:*
    - *Contains information about threads: thread id, thread context, thread status, thread priority, thread run count and thread return pointer.*

- *thread_node struct:*
    - *For Thread node context, which contains pointer to tcb and a next pointer to Thread_node.*

- *queue struct:*
    - *For Queues which contains pointers to front and rear of the queue of thread nodes, and also the queue size. Front and rear pointers will point to the Thread_node struct.*

- *mypthread_mutex_t struct:*
    - *For Mutex lock and unlock. Also contains a waiting queue for threads blocked on Mutex.*

- *scheduler struct:*
    - *For Scheduler, which contains pointers to the ready queue, main thread tcb and current thread tcb.*

Functions:

Declarations for thread creation and management, scheduling and the mutex function calls:

- *int mypthread_create(mypthread_t * thread, pthread_attr_t * attr, void *(*function)(void*), void * arg);*
    - *Creates a new thread and it gets added to the respective queue by the scheduler.*

- *int mypthread_yield();*
    - *Current thread will surrenders its remaining runtime for other threads to use.*

- *void mypthread_exit(void *value_ptr);*
    - *Terminates a thread and swaps context to main.*

- *int mypthread_join(mypthread_t thread, void **value_ptr);*
    - *Wait for a thread to terminate.*

- *int mypthread_mutex_init(mypthread_mutex_t *mutex, const pthread_mutexattr_t *mutexattr);*
    - *Initializes a mutex object.*

- *int mypthread_mutex_lock(mypthread_mutex_t *mutex);*
    - *Acquire a mutex (lock) using test and set atomic operation.*

- *int mypthread_mutex_unlock(mypthread_mutex_t *mutex);*
    - *Release a mutex (unlock)*

- *int mypthread_mutex_destroy(mypthread_mutex_t *mutex);*
    - *Destroy a mutex.*

Declarations for scheduler function calls:

- *void schedule();*
    - *To choose which scheduling algorithm to run.*

- *static void sched_RR()*
    - *time slices (also known as time quanta) are assigned to each thread in equal portions and in circular order, handling all processes without priority.*

- *static void sched_PSJF()*
    - *In the Shortest Remaining Time First (SRTF) scheduling algorithm, the process with the smallest amount of time remaining until completion is selected to execute.*

- *static void sched_MLFQ()*
    - *Threads in the ready state are divided into different groups with priority, each group having its own scheduling needs.*

- *void execute_thread(tcb *tcb_ptr, void *(*f)(void *), void * arg);*
    - *To run thread using parameters from arg. The run thread takes a thread and executes the function passed with the passed arguments.*

- *void scheduler_init();*
    - *Initiate scheduler when the first thread is created.*
    - *Creates priority queues and initialize a queue at each index.*

- *void ready_thread(tcb *tcb_ptr, int priority);*
    - *Adds a thread to the ready queue.*

- *tcb *select_thread();*
    - *For choosing a thread from number of priority queues.*

- *void timer_init();*
    - *Initializes the alarm and time quantum.*

- *void sighandler();*
    - *Used for signal handling.*

Declarations for queue function calls:

- *void queue_init(queue *q);*
    - *Initializes a queue.*

- *thread_node *thread_node_init();*
    - *Contains pointer to the tcb and next pointer to a thread_node in a queue.*

- *void enqueue(queue *first, tcb *thread_node);*
    - *Adds queue node to the queue*

- *tcb *dequeue(queue * q);*
    - *Returns the front of the queue.*

3 Benchmark Results

MLFQ:

For Parallel_cal

No. of threads	*Mypthread (Micro Seconds)*	*Pthread (Micro Seconds)*
2	*6928*	*3099*
20	*6852*	*617*
50	*6880*	*200*
70	*3895*	*167*
99	*3790*	*140*
	Sum = 83842816	*Sum = 83842816*
	Verified sum= 83842816	*Verified sum= 83842816*

For Vector_Multiply

No. of threads	*Mypthread (Micro Seconds)*	*Pthread (Micro Seconds)*
2	*33*	*183*
20	*48*	*320*
50	*95*	*433*
70	*113*	*555*
99	*145*	*610*
	Sum = 631560480	*Sum = 631560480*
	Verified Sum = 631560480	*Verified Sum = 631560480*

RR:

For Parallel_cal

No. of threads	*Mypthread (Micro Seconds)*	*Pthread (Micro Seconds)*
2	*2954*	*3081*
20	*2793*	*302*
50	*2960*	*199*
70	*3187*	*174*
99	*4592*	*156*
	Sum =83842816	*Sum =83842816*
	Verified sum =83842816	*Verified sum =83842816*

For Vector_Multiply

No. of threads	*Mypthread (Micro Seconds)*	*Pthread (Micro Seconds)*
2	*33*	*190*
20	*47*	*378*
50	*60*	*524*
70	*54*	*592*
99	*62*	*652*
	Sum =631560480	*Sum =631560480*
	Verified sum =631560480	*Verified sum =631560480*

PSJF:

For Parallel_cal

No. of threads	*Mypthread (Micro Seconds)*	*Pthread (Micro Seconds)*
2	*6710*	*732*
20	*6391*	*326*
50	*6182*	*227*
70	*5954*	*212*
99	*6156*	*211*
	Sum =83842816	*Sum =83842816*
	Verified sum =83842816	*Verified sum =83842816*

For Vector_Multiply

No. of threads	*Mypthread (Micro Seconds)*	*Pthread (Micro Seconds)*
2	*33*	*194*
20	*45*	*332*
50	*91*	*529*
70	*117*	*866*
99	*123*	*1116*
	Sum =631560480	*Sum =631560480*
	Verified sum =631560480	*Verified sum =631560480*