PREPOSTSEO

PLAGIARISM COMPARISON SCAN REPORT

Content Type	TEXT	TEXT
Values	Text content used	Text content used

First Content **4%** matched	Second Content **4%** matched

include bitsinclude pthreadinclude unistdinclude sysinclude sysinclude sysinclude sysusing namespace stddefine MAX 100000000define MAXJOBS 10000define MAXCHILD 100define MAXTHREADS 20define endl nclass Job public int jobId, timeForCompletion, status, dependentJobs[MAXCHILD], childCount 0 emptyprevious completed, 1 created, 2 running Job() jobId rand()%MAX 1 timeForCompletion rand()%1 status 0 childCount 0 for(int i0iMAX dependentJobs[i]-1 nodeCount0 xssremoved xssremoved xssremoved lifetime1000 currtime0lock) vectorint available for(int i0iMAXjobs[i].status0) break if(tree-jobs[i].status1 tree-jobs[i].childCountMAX xssremovedlock) break int jobToSelect available[rand()%((int)(available.size()))] Job j Job() j.status1 int indexToPut-1 for(int i0iMAXjobs[i].status0) indexToPuti break tree-jobs[indexToPut]j int childIndex-1 for(int i0iMAXjobs[jobToSelect].dependentJobs[i]-1) childIndexi break tree-jobs[jobToSelect].dependentJobs[childIndex]indexToPut tree-jobs[jobToSelect].childCount tree-nodeCount coutltltNew Job Created with Job Id ltjlock) int t rand()P1 if(usleep(t1000)-1) coutltltSleep Error Detected.ltendl currtimetlock) if(tree-nodeCountlt0) pthreadmutexunlock(tree-lock) break int index-1 for(int i0iMAXjobs[i].status1) if(tree-jobs[i].childCount0) indexi break if(index-1) pthreadmutexunlock(tree-lock) break tree-jobs[index].status2 int ttree-jobs[index].timeForCompletion coutltltJob Started with Job Id lttreejobs[index].jobIdltendllock) if(usleep(t1000)-1) coutltltError in sleep.ltendllock) coutltltJob Completed with Job Id lttreejobs[index].jobIdltendljobs[index].status0 for(int i0iMAXjobs[i].status1) continue for(int j0jMAXjobs[i].dependentJobs[j]index) swap(tree-jobs[i].dependentJobs[j], tree-jobs[i].dependentJobs[tree-jobs[i].childCount-1]) tree-jobs[i].dependentJobs[tree-jobs[i].childCount-1]-1 tree-jobs[i].childCount-- break tree-nodeCount-- pthreadmutexunlock(tree-lock) pthreadexit(0)int main() pthreadt producerThreads[MAXTHREADS], consumerThreads[MAXTHREADS] Shared Memory Creation keyt key ftok(devrandom,'c') int shmid shmget(IPCPRIVATE,sizeof(Tree),0666IPCCREAT) Storing Tree in Shared Memory tree (Tree) shmat(shmid, NULL, 0) mutex locks initiation pthreadmutexattrt lockattr pthreadmutexattrinit(lockattr) pthreadmutexattrsetpshared(lockattr, PTHREADPROCESSSHARED) pthreadmutexinit(tree-lock, lockattr) for(int i0iMAXjobs[i]Job() int create300rand() 1 int create3rand()/or(int i0icreatejobs[i].status1 coutltltNew Job Created with Job Id lttreejobs[i].jobIdltendlnodeCountcreate int producers, consumers coutltltEnter

the number of$_3$

Producers cinproducers coutltltEnter

the number of$_3$

Consumers cinconsumers pthreadattrt attr pthreadattrinit(attr) for(int i0iltproducersi) Create threads for producers pthreadcreate(producerThreads[i], attr, producer, NULL) if(fork()0) for(int i0iltconsumersi) Create consumers pthreadcreate(consumerThreads[i], attr, consumer, NULL) for(int i0iltconsumersi) pthreadjoin(consumerThreads[i], NULL) wait for each thread to end

shmdt(tree) exit(0) else for(int$_2$

i0iltproducersi) pthreadjoin(producerThreads[i], NULL) wait for each thread to complete

wait(NULL) shmdt(tree) shmctl(shmid, IPCRMID, 0) return 0$_1$

include bitsinclude pthreadinclude sysinclude unistdinclude sysusing namespace stddefine MAXJOBS 30000define MAXDEPENDENTJOBS 100define MAXTHRDS 20struct job int jobid int timeofcompletion int status -1 - empty, 0 - just created , 1- ongoing , 2- completed int numdependent int dependentjobs[MAXDEPENDENTJOBS] pthreadmutext lockstruct Tree job jobs[MAXJOBS] pthreadmutext lock int numjobsTree treepthreadt producerthreads[MAXTHRDS], consumerthreads[MAXTHRDS]job createjob() job j j.jobid 1 rand()0000000 j.timeofcompletion rand()%1 j.status 0 for(int i 0 i MAX xssremoved xssremoved xssremoved xssremoved xssremovedlock) vectorint indices for(int i 0 i MAXjobs)[i].status 0) indices.pushback(i) if((int)indices.size() 0) pthreadmutexunlock(tree-lock) break int toput indices[rand()%(indices.size())] job j createjob() for(int i 0 i MAXjobs)[toput].dependentjobs[i] -1) continue (tree-jobs)[toput].dependentjobs[i] j.jobid (tree-jobs)[toput].numdependent 1 break for(int i 0 i MAXjobs)[i].status -1) (tree-jobs)[i] j break tree-numjobs 1 cout ltlt New job created lt jlock) int sleeptime rand()P1 if(usleep(sleeptime1000) -1) printf(Error in sleepn) exit(1) curtime sleeptime pthreadexit(0)void consumerrunner(void param) while(1) pthreadmutexlock(tree-lock) if((tree-numjobs) lt 0) pthreadmutexunlock(tree-lock) break int jobtoexecute -1 for(int i 0 i MAXjobs)[i].status 0 (tree-jobs)[i].numdependent 0) jobtoexecute i break if(jobtoexecute -1) pthreadmutexunlock(tree-lock) break (tree-jobs)[jobtoexecute].status 1 cout ltlt Start of job ltlt (tree-jobs)[jobtoexecute].jobid lt endllock) int timetosleep (tree-jobs)[jobtoexecute].timeofcompletion if(usleep(timetosleep1000) -1) printf(Error in sleepn) exit(1) pthreadmutexlock(tree-lock) cout ltlt End of job ltlt (tree-jobs)[jobtoexecute].jobid lt endljobs)[jobtoexecute].status -1 for(int i 0 i MAX xssremovedjobs)[i].status 0) for(int j 0 j MAXjobs)[i].dependentjobs[j] (tree-jobs)[jobtoexecute].jobid) (tree-jobs)[i].dependentjobs[j] -1 (tree-jobs)[i].numdependent - 1 break (tree-numjobs) - 1 pthreadmutexunlock(tree-lock) pthreadexit(0)int main() keyt key 235 int shmid shmget(key, sizeof(Tree), IPCCREAT0666) if(shmid 0 xssremoved xssremovedjobs)[i].status -1 int numjobs 300 rand() 1 int numjobs 3 rand()%3 tree-numjobs numjobs for(int i 0 i numjobs)[i] createjob() coutltltJob Created with Job id lttreejobs[i].jobidltendllock, lockattr) int P cout ltlt Enter

the number of$_3$

producer threads cin P int C cout ltlt Enter

the number of$_3$

consumer threads cin C pthreadattrt attr pthreadattrinit(attr) for(int i 0 i lt P i) pthreadcreate(producerthreads[i], attr, producerrunner, NULL) pidt pid pid fork() if(pid 0) for(int i 0 i lt C i) pthreadcreate(consumerthreads[i], attr, consumerrunner, NULL) for(int i 0 i lt C i) pthreadjoin(consumerthreads[i], NULL)

shmdt(tree) exit(0) else for(int$_2$

i 0 i lt P i) pthreadjoin(producerthreads[i], NULL)

wait(NULL) shmdt(tree) shmctl(shmid, IPCRMID, 0) return 0$_1$