Grupo 1: Chrsytian de Sousa Guth
Lucas Pereira da Silva
Renan Oliveira Netto

Problema:

You are requested to modify the implementation of Thread so that join() no longer wastes time sampling the waited thread status. It must be set to sleep until the waited-for thread terminates, and then waken up accordingly.

Solução:

A solução utiliza uma fila de threads esperando e métodos que adicionam e removem threads desta fila.

Para isso utilizou-se o atributo _waiting da classe Thread representando as threads que estão esperando pela dona do atributo. Este atributo já existia na classe Thread, mas não estava sendo alocado um novo espaço de memória para ele, por isso foi necessário a adição de uma nova instrução no construtor responsável por isso.

O método join verifica se a thread que recebeu a chamada já terminou ou não sua execução. Caso já tenha terminado, o método apenas retorna, caso ainda não tenha terminado, a método chama sleep, passando como parâmetro o atributo _waiting da thread que recebeu a chamada. Isso é feito para que a thread que chamou o método join seja inserida na fila _waiting.

O método sleep por sua vez obtém a referência da thread em execução, suspende ela para que a mesma não seja escolhida pelo escalonador, muda seu estado para WAITING, adiciona ela na fila recebida como parâmetro e por fim chama o método dispatch, que muda o contexto da thread atual para uma nova thread escolhida pelo escalonador. Como sleep é chamado por join recebendo a fila _waiting como parâmetro, isso equivale a suspender a thread em execução e adicioná-la na fila _waiting da thread que recebeu o join.

Para acordar as threads que estiverem na fila _waiting foi adicionada uma instrução chamando o método wakeup_all no método exit da classe Thread. Essa instrução passa como parâmetro a fila _waiting da thread em execução (a thread que está terminando), de forma que o método wakeup_all acorde todas as threads da fila _waiting.

Por fim o método wakeup_all percorre a fila recebida como parâmetro, remove cada elemento da mesma, muda seu estado para READY e chamada o método resume do escalonador, que avisa o mesmo para voltar a considerar a thread t no escalonamento. Desta forma, todas as threads que estavam na fila _waiting da thread que está encerrando voltam a ser escalonadas.

Métodos modificados na classe Thread:

Método common_constructor:

```
void Thread::common_constructor(Log_Addr entry, unsigned int stack_size)
{
   db<Thread>(TRC) << "Thread(entry=" << (void *)entry
                   << ",state=" << _state
                   << ",rank=" << _link.rank()
                   << ",stack={b=" << _stack
                   << ",s=" << stack_size
                   << "},context={b=" << _context
                   << "," << *_context << "}) => " << this << "\n";

   _thread_count++;

        // Inicializa a fila _waiting
        _waiting = new Queue();

   _scheduler.insert(this);
   if((_state != READY) && (_state != RUNNING))
      _scheduler.suspend(this);

   unlock();

}
```

Método join:

```
int Thread::join()
{
   db<Thread>(TRC) << "Thread::join(this=" << this
                   << ",state=" << _state << ")\n";

        /* Verifica se a thread já não terminou e, caso não tenha terminado,
                coloca a thread em execução para dormir
        */
        if (_state != FINISHING)
                sleep(this->_waiting);

   return *static_cast<int *>(_stack);
}
```

Método sleep:

```
void Thread::sleep(Queue * q)
{
        // Obtém a thread em execução
   Thread * thr = running();

        /* Suspende a thread em execução,
                muda seu estado para waiting
                e a adiciona na fila q

        */
```

```
    _scheduler.suspend(thr);
    thr->_state = WAITING;
    q->insert(&thr->_link);

        // Chama dispatch para executar a thread seguinte
    dispatch(thr, _scheduler.chosen());
}
```

Método wakeup_all:

```
void Thread::wakeup_all(Queue * q)
{
        /* Percorre a fila q,
                removendo cada elemento da fila,
                mudando seu estado para READY
                e chamando o método resume do escalonador
        */
    while(!q->empty()) {
        Thread * t = q->remove()->object();
        t->_state = READY;
        _scheduler.resume(t);
    }
}
```

Método exit:

```
void Thread::exit(int status)
{
    lock();

    db<Thread>(TRC) << "Thread::exit(running=" << running()
                    <<",status=" << status << ")\n";

    Thread * thr = running();
    _scheduler.remove(thr);
    *static_cast<int *>(thr->_stack) = status;
    thr->_state = FINISHING;

        // Acorda as threads que estiverem na fila _waiting da thread que está terminando
        wakeup_all(thr->_waiting);

    _thread_count--;


    dispatch(thr, _scheduler.choose());
}
```