UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL
INSTITUTO DE INFORMÁTICA
DEPARTAMENTO DE INFORMÁTICA APLICADA

Implementação da biblioteca μthread - Trabalho prático I

1. Nome dos componentes do grupo e número do cartão.
Pedro Eduardo Bahi de Souza 122574
Mike Muya 213428

2. Descrição da plataforma utilizada para desenvolvimento.
Qual o tipo de processador (número de cores, com ou sem suporte HT) ?
Quatro cores com suporte a HT.

Qual a distribuição GNU/Linux utilizada e a versão do núcleo ?
Linux guasco-VirtualBox 2.6.38-8-generic #42-Ubuntu SMP Mon Apr 11 03:31:50 UTC 2011 i686 i686 i386 GNU/Linux

Qual a versão do gcc ?
gcc version 4.5.2 (Ubuntu/Linaro 4.5.2-8ubuntu4)

Se o trabalho foi feito ou não em ambientes virtualizados ? Em caso afirmativo, qual a máquina virtual utilizada (versão) ?
Oracle Virtual Box 4.1.2r73507

3. Para cada programa de teste elaborado pelo grupo : descrever o que programa faz ; indicar claramente quais os parâmetros a serem passados para sua execução e qual a saída esperada.

testes_threads: Incializa e executa 15 threads, que informan quando inciam, cedem cpu ou terminam, as threads executam computações e a thread 6 precisa aguardar o termino da thread 8.

4. Explique o funcionamento da primitiva thread_create desenvolvida pelo grupo, citando as principais estruturas de dados envolvidas e funções chamadas.

A thread_create recebe como parâmetros uma função e um ponteiro para seus argumentos, após é alocado a estrutura que conterá o contexto, pilha e id, então atraves da função SetReady, a thread é colocada em na fila de Ready.

Implementação da biblioteca μthread - Trabalho prático I

5. Explique o funcionamento da primitiva thread_yield desenvolvida pelo grupo, citando as principais estruturas de dados envolvidas e funções chamadas.

A principal estrutura envolvida é uma variavel publica chamada Running_thread que aponta para a thread que está executando.

O funcionamento é mudar o estado da thread atual para Ready e devolver o controle para o escalonador com swapcontext.

6. Explique o funcionamento da primitiva thread_join desenvolvida pelo grupo, citando as principais estruturas de dados envolvidas e funções chamadas.

A join coloca a thread que fez a chamada na lista de bloqueada, remove ela da lista de Ready, então fica executando até que que a thread alvo termine.

As principais estruturas são a thread que se espera que termine (thread_that_must_finish) a fila de Ready e a chamada da função interna run_thread, que executa threads a partir do ponto em que pararam até que terminem ou devolvam a cpu.

7. Descrever o que funciona na thread desenvolvida e o que NÃO está funcionando. Em caso de não funcionamento, dizer qual é a sua visão do porquê deste não funcionamento.

Acreditamos que todas funcionalidades foram desenvolvidadas, fizemos apenas uma alteração diferente do que foi especificado buscando um melhor desempenho. A função thread create não está retornando int, mas um ponteiro para uma estrutura de thread, foi implementado dessa maneira para evitar de percorrer as filas para descobrir a thread de id especifico, o que consumiria um tempo desnecessário, também foi observado que a função join exigia um parâmetro de ponteiro para thread, para que isso fosse possível teria que se criar uma função não especificada para recuperar a thread conforme um id. Então se entendeu poderia ser uma melhoria a especificação, mesmo considerando isso um risco.

8. Qual a metodologia de teste utilizada ? Isto é, quais foram os passos (e programas) efetuados para testar a thread desenvolvido ? Foi utilizado um debugger ? Qual ?

Foram desenvolvidas funções de debug para as estruturas utilizadas, a ferramenta de debug utilizada foi o gdb somente para verificação de segmentation fault, sendo que não tivemos muito problema desse tipo. Primeiro preparamos um repositório para controle de verão visando agilizar o trabalho em grupo. Após estudamos a documentação oficial da context.h e nos baseamos no exemplo fornecido, então foram criadas funções de debug e por fim um escalonador basico e

estático, que era capaz de executar threads até que terminassem sem a possibilidade de yeld. A partir desse momento o entendimento da biblioteca foi dominado e todas funções de escalonador foram concluída, e por ultimo foi implementada um controle de filas.

9. Quais as principais dificuldades encontradas e quais as soluções empregadas para contorná-las.

As principais dificuldades foi a nomenclatura, os termos utilizados pela biblioteca por mais que agora sejam obvios e evidentes, alem de bastante abordados em sala de aula são bem difíceis de entender a primeira vez, após esse processo percebe-se que as funções são extremamente óbvias e que a expectativa de algo complexo e difícil é o principal entrave.
Após o dominio completo das funções, se acreditou que o trabalho estava quase concluido, é quando se percebeu que para que funções como join e yield funcionem corretamente não basta uma simples aplicação de funções prontas, é necessário traçar estratégias de funcionamento e fazer escolhas, que afetarão a estabilidade, isolamento e desempenho do trabalho. Por fim outra principal dificuldade foi coordenar e dividir a realização o trabalho em grupo.