INSTITUTO DE INFORMÁTICA
UFRGS

Sistemas Operacionais I - **µ**núcleo

Pedro Pillon Vanzella [213990] <pedro@pedrovanzella.com>

Relatório de Atividades

1 - Componentes do Grupo

Pedro Pillon Vanzella [213990]

2 - Plataforma

O trabalho foi desenvolvido no OSX, por também ser um sistema Unix, baseado no FreeBSD e, supostamente, prover todos os recursos que o Linux proveria. A máquina na qual o trabalho foi desenvolvido tem um processador Intel Core i7 950, com 4 núcleos a 3.06GHz, com suporte HT. A versão instalada do gcc é a i686-apple-darwin11-llvm-gcc-4.2 (GCC) 4.2.1 (Based on Apple Inc. build 5658) (LLVM build 2336.9.00). O trabalho foi desenvolvido, porém, com o clang, por este dar melhores mensagens de erro e avisos. Graças a um makefile bem feito, foi possível facilmente trocar para o gcc e verificar que o trabalho compila corretamente com ele. A versão do clang é Apple clang version 3.1 (tags/Apple/clang-318.0.54) (based on LLVM 3.1svn). Durante todo o desenvolvimento, foi utilizado a ferramenta git para controle de versão, provendo tanto um backup quanto um histórico de modificações do código.

3 - Programas de Teste

exemplo1.c

Um exemplo bem simples, que simplesmente verifica que a troca de contextos funciona. Imprime “Hello, World!”.

exemplo2.c

Verifica que temos, de fato, uma FIFO. Uma função imprime os números de 1 a 5 e, quando esta termina, a outra imprime os números de 6 a 9.

parouimpar.c

Imprime, com uma função, os números pares, e, com a outra, os números ímpares, alternando entre elas através da diretiva yield da biblioteca.

4 - mproc_create()

A função mproc_create() cria um novo contexto, através da makecontext(), insere em um pcb e o insere no final da lista de aptos.

5 - mproc_yield()

A função mproc_yield() procura um contexto na lista de aptos, respeitando a regra de prioridade, insere o processo atual no fim da lista de aptos e troca para o contexto do processo encontrado.

6 - mproc_join()

A função mproc_join() não foi feita. Simplesmente retorna 1.

7 - Resultados

O que funciona corretamente:

A criação de PCBs, o enfileramento dos mesmos, a busca de processos, a criação de novos processos funcionam corretamente.

O que não funciona:

O dispatcher parece não iniciar, impedindo o funcionamento da biblioteca por completo. Isto pode ser culpa do depreciamento da ucontext.h pelo OSX. O programa precisa que o símbolo _XOPEN_SOURCE seja definido para que possa ser compilado, e todas as funções da ucontext.h estão marcadas como depreciada. Exemplos vistos em aula não funcionam no OSX por causa disso. Muito provavelmente estas funções foram depreciadas (e não são mais mantidas desde 2002 graças a isso) por causa da introdução do GCD no OSX, que é uma nova maneira de gerenciamento de threads, e seria quebrado caso algum programa fizesse manipulação explícita de contexto.

8 - Testes

Os três primeiros programas de teste (exemplo1, exemplo2 e par_ou_impar) foram feitos antes mesmo de qualquer código (fora alguns *stubs* para que compilassem corretamente) fossem escritos. O debugger gdb foi utilizado no início, mas tornou-se pouco prático, dada a alternativa escolhida: colocar printf()s em pontos estratégicos do código.

9 - Dificuldades

O trabalho foi feito individualmente, contrário à sugestão do professor. Isto se provou um pequeno desafio, dado o tamanho do trabalho.

Do ponto de vista técnico, foi necessário mover a inicialização das listas de apto e bloqueado para o dispatcher, pois eles necessitavam acesso direto a elas.

O trabalho também foi levemente subestimado, o que resultou em um dispatcher que não inicializa e uma função mproc_join() incompleta.