Iniciado em	Sunday, 1 May 2022, 21:10
Estado	Finalizada
Concluída em	Sunday, 1 May 2022, 21:12
Tempo empregado	1 minuto 54 segundos
Avaliar	**9,3** de um máximo de 10,0(**93**%)

Questão 1

Correto

Atingiu 1,0 de 1,0

1. A inicialização correta de um processo criado para executar um programa escrito em C++ que possui objetos globais com construtores não vazios pressupõe que os mesmos sejam inicializados antes da função *main()*. Para tal, estes construtores de objetos globais são emitidos pelo compilador na seção ELF [.init] e sua ordem de invocação é definida pelo [linker].

2. O *placement new* (i.e. *inline void * operator new(size_t s, void * a) { return a; }*) é utilizado para [construir um objeto em uma posição de memória previamente definida].

Questão 2

Parcialmente correto

Atingiu 4,3 de 5,0

Para as questões a seguir considere a versão idle thread do código do EPOS no GIT.

1. Utilizando gdb responda as seguintes questões sobre a aplicação *philosophers_dinner* executando sobre uma máquina *RV32/RISCV/SiFive_E*:

- Qual o valor do stack pointer no momento em que o sistema entra em idle() a primeira vez? 87ff6f64
- Analisando o endereço do método idle, é possível afirmar que ele se localiza em um endereço [menor] que o stack pointer.
- Neste ponto, as interrupções da CPU estão [habilitadas].
- É possível afirmar que [o sistema só executa a thread idle após todos as tarefas estarem finalizadas].

2. Crie uma nova aplicação no EPOS, executando sobre uma máquina *RV32/RISCV/SiFive_E* com uma CPU que executa o código abaixo. Sobre a execução do código, responda as seguintes questões (dica: crie contadores em pontos específicos da execução):

```
#include<time.h>
#include <synchronizer.h>
#include <process.h>

using namespace EPOS;
const int iterations = 10;
OStream cout;
Semaphore lets_call_idle(0);

void release_lock() {
   lets_call_idle.v();
}

int main() {
   cout << "Simple Idle Test" << endl;
   Function_Handler fh(&release_lock);
   Alarm a(1000000,&fh, iterations);
   for (int i = 0; i < iterations; i++) {
       lets_call_idle.p();
       cout << "Main was Idle " << i+1 << " times" << endl;
   }

   cout << "The end!" << endl;
   return 0;
}
```

- A função main chama a função de bloqueio [10] vezes.
- Acontecem durante a execução [mais que 10] dispatches.
- A thread Idle executa halt [mais que 1000] vezes.

Questão 3

Correto

Atingiu 4,0 de 4,0

1. Uma das definições feitas no EPOS para criar a thread *idle* foi definir [uma nova prioridade] , o que possibilita controlar o uso da CPU via uma ordenação de threads, na qual a thread de mais alta prioridade possui o [menor] valor e executa [primeiro] .

2. Utilizando-se a implementação de *idle* como uma thread, não é mais necessário executar-se verificações sobre a fila [ready] , pois [a fila de threads sempre conterá, no mínimo a idle] .

3. O [atributo _thread_count] é utilizado como condição de finalização do sistema, e seu valor para que isso aconteça em uma configuração singlecore com a *idle* modelada como thread deve ser [=1] . Num cenário de sistema operacional multicore, o número de threads restantes (contando com as threads idle) no sistema para que a *idle* não ordene o desligamento [deve ser maior que o número de CPUs] . (CPU = core lógico)

4. A troca de *idle* de um método que é chamado em momentos de ociosidade para uma thread implicou na remoção de várias verificações. Em relação a isto, não é correto afirmar-se que a mudança acarretou [aumento na facilidade de implementação do código da aplicação] .

5. O EPOS inicializa o sistema de threads criando a thread [main] como *running*, e então carrega seu contexto [depois] de criar a thread [idle] , sendo que a última não é atribuída a um atributo [porque ela é inserida pelo construtor na fila _ready] .

6. Qual o último elemento da tabela de inicialização dos construtores globais do EPOS? [init end] .

◄ OSDI with EPOS: Idle Thread

Seguir para...

E3b: OSDI with EPOS: Console Output (coding) ►