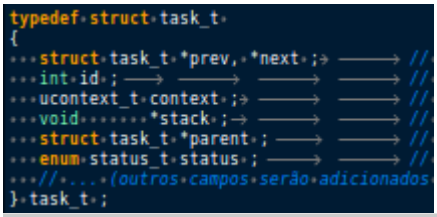
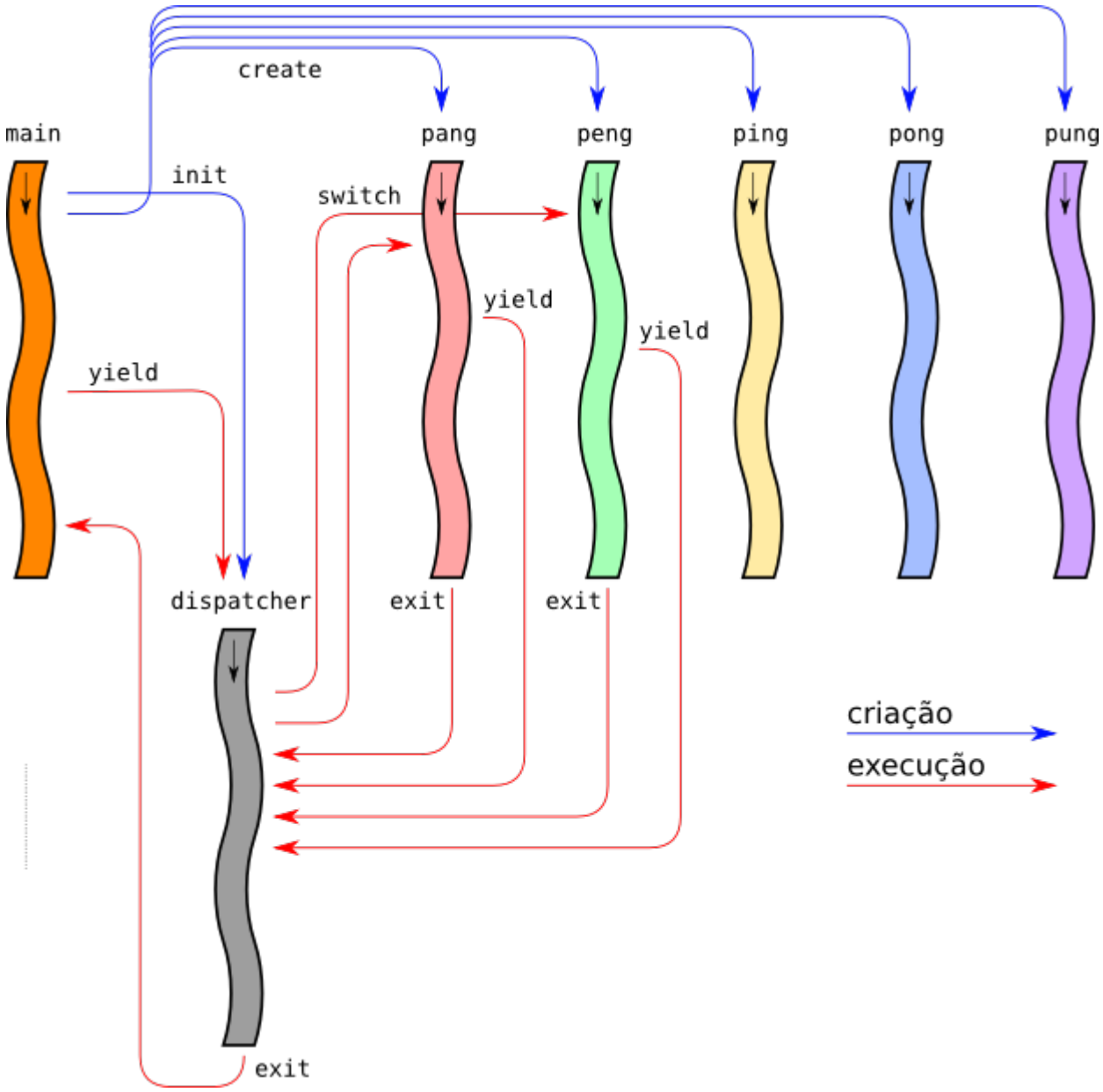


Carlos Maziero

Despachante de tarefas

Você irá construir um despachante de tarefas baseado em duas entidades: uma tarefa *dispatcher*, responsável pelo controle geral, e uma função *scheduler*, responsável por determinar qual a próxima tarefa a executar a cada troca de contexto. A figura abaixo ilustra o funcionamento geral do sistema:



A seguinte chamada permite a uma tarefa voltar ao final da fila de prontas, devolvendo o processador ao *dispatcher*:

```
void task_yield () ;
```

Use `task_switch` para implementar `task_yield`.

Observações

- O *dispatcher* deve ser implementado como uma tarefa, a ser criada usando a chamada `task_create` durante a inicialização do sistema (execução de `pingpong_init`).
- O programa principal cria todas as tarefas de usuário e passa o controle para a tarefa *dispatcher*, que só encerra quando não existirem mais tarefas de usuário a executar.
- Será necessário implementar uma fila de tarefas prontas, usando a biblioteca de filas genéricas desenvolvida anteriormente.
- Quando uma tarefa encerra (usando `task_exit`), o controle volta ao *dispatcher* e as estruturas de dados usadas pela tarefa são liberadas.
- A **política de escalonamento** será definida por uma função `scheduler()`, chamada pelo *dispatcher* para decidir qual a próxima tarefa a ativar. Neste projeto, deve ser implementada uma política FCFS.

O código do corpo da tarefa *dispatcher* deve seguir +/- o seguinte modelo (simplificado):

```
dispatcher_body () // dispatcher é uma tarefa
{
   while ( userTasks > 0 )
   {
      next = scheduler() ;  // scheduler é uma função
      if (next)
      {
         ... // ações antes de lançar a tarefa "next", se houverem
         task_switch (next) ; // transfere controle para a tarefa "next"
         ... // ações após retornar da tarefa "next", se houverem
      }
   }
   task_exit(0) ; // encerra a tarefa dispatcher
}
```

Sua implementação deve funcionar com este código. A saída da execução deve ser igual a este exemplo.

Chamadas suspend/resume

As chamadas seguintes não são usadas pelos programas de teste atuais, mas simplificam a implementação dos projetos futuros:

Suspende uma tarefa

```
void task_suspend (task_t *task, task_t **queue)
```

Suspende uma tarefa, retirando-a de sua fila atual, adicionando-a à fila `queue` e mudando seu estado para "suspensa". Se `task` for nulo, considera a tarefa corrente. Se `queue` for nulo, não retira a tarefa de sua fila atual.

Acorda uma tarefa suspensa

```
void task_resume (task_t *task)
```

Acorda uma tarefa, retirando-a de sua fila atual (se estiver em uma fila), adicionando-a à fila de tarefas prontas (*ready queue*) e mudando seu estado para "pronta".

Outras informações

- Duração estimada: 5 horas.
- Dependências:
  - Biblioteca de Filas
  - Gestão de Tarefas

---

so/dispatcher.txt · Última modificação: 2015/04/09 11:52 por maziero