Sistemas Operacionais
Processos / Threads

Carlos Ferraz (cagf@cin.ufpe.br)
Jorge Cavalcanti Fonsêca (jcbf@cin.ufpe.br)

Disciplina

- Facebook
  - CIn - Sistemas Operacionais - 2015.1
  - https://www.facebook.com/groups/1425133897791021/

- Lista de Monitoria
  - ??? (Vou falar com o monitor)

Disciplina

Index of /~jcbf/if677/2015-1/slides

Name	Last modified	Size	Description
Parent Directory		-	
Aula_01_02_03_Introducao_SO.pdf	17-Apr-2015 08:00	2.7M	
Aula_04_Processos.pdf	17-Apr-2015 07:55	916K	
Aula_04_Processos_Uso_Fork.c	17-Apr-2015 07:58	1.1K	

www.cin.ufpe.br/~cagf/if677/2015-1/

Bibliografia:

Sistemas Operacionais Modernos – 2ª Edição. A. Tanenbaum, Prentice-Hall, 2003.

Opção: *Modern Operating Systems* 3e. A. Tanenbaum, Prentice-Hall, 2008 (Já em Português, edição 2010).

Calendário (Slides disponíveis aqui)

No.	Data	Aula	Responsável
01	18/3	Visão Geral de Sistemas Operacionais	Carlos
02	20/3	Visão Geral e Estrutura de Sistemas Operacionais	Carlos
	25-27/3	PerCom 2015	
03	1/4	Estrutura de Sistemas Operacionais	Carlos
Em viagem: 06/04-05/07/2015			Carlos
04	10/4	Processos, Interrupções e Chamadas ao Sistema	Jorge
05	17/4	Processos / Threads	Jorge
06	24/4	Threads e POSIX Threads (PThreads)	Jorge
X	01/5	FERIADO	
07	08/5	Concorrência: Condição de Disputa, Região Crítica, Exclusão Mútua, Semáforos e Monitores	Jorge
08	13/5 (quarta)	Especificação da Lista de Threads	Monitoria
09	15/5	Concorrência (cont.) e *Deadlock*	Jorge

Copyright

Carlos Ferraz

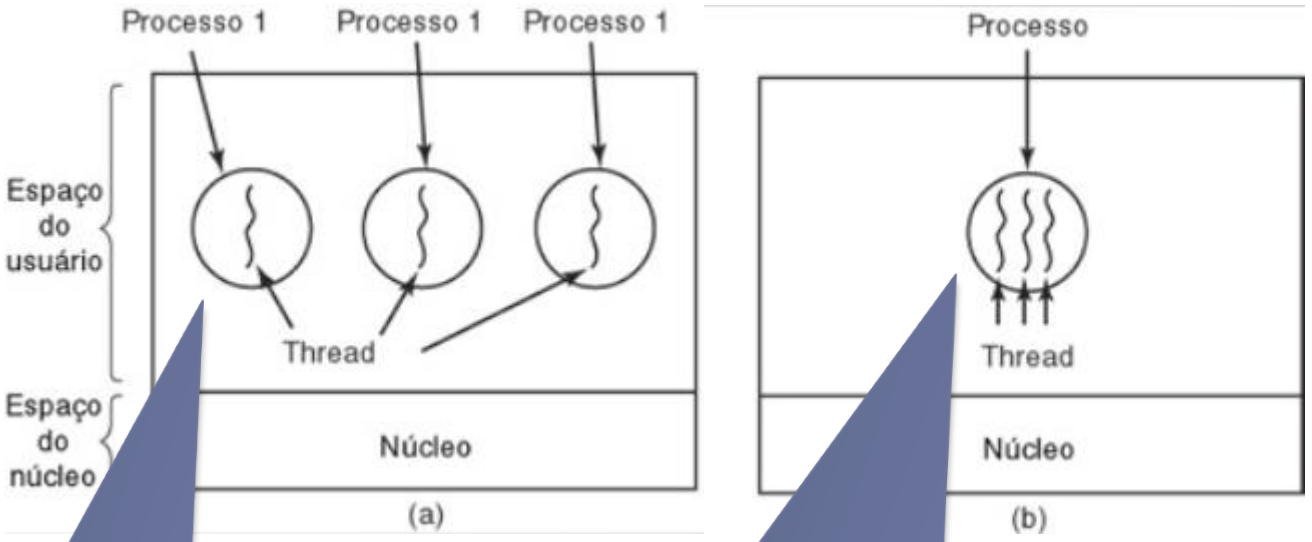
Processo

```c
#include <stdio.h>

int main (int argc, char** argv)
{
    printf("Hello World!\n");
    return (0);
}
```

1 processo – 1 Thread (principal)

O que é um(a) thread?

- Linha de execução dentro um processo.



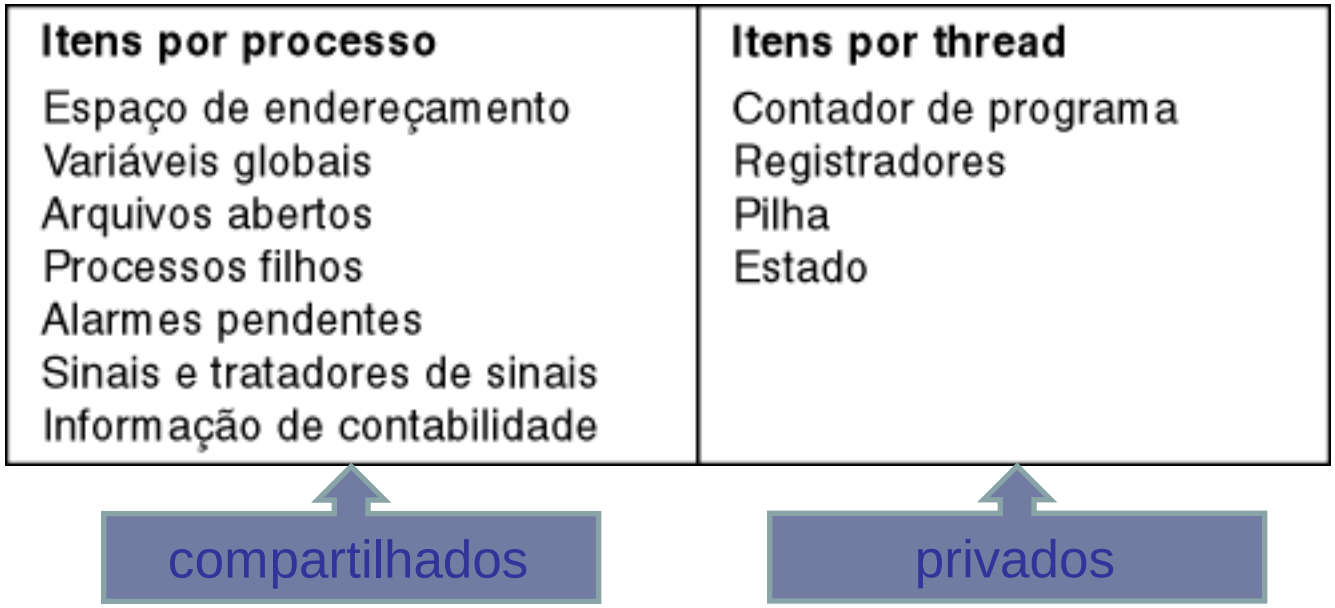
Cada processo tem um espaço de endereçamento e uma única linha (*thread*) de controle/execução

No entanto, há várias situações em que é desejável ter múltiplas linhas de controle no mesmo espaço de endereçamento rodando em "paralelo" como se fossem processos (leves) separados (exceto pelo espaço de endereçamento compartilhado)

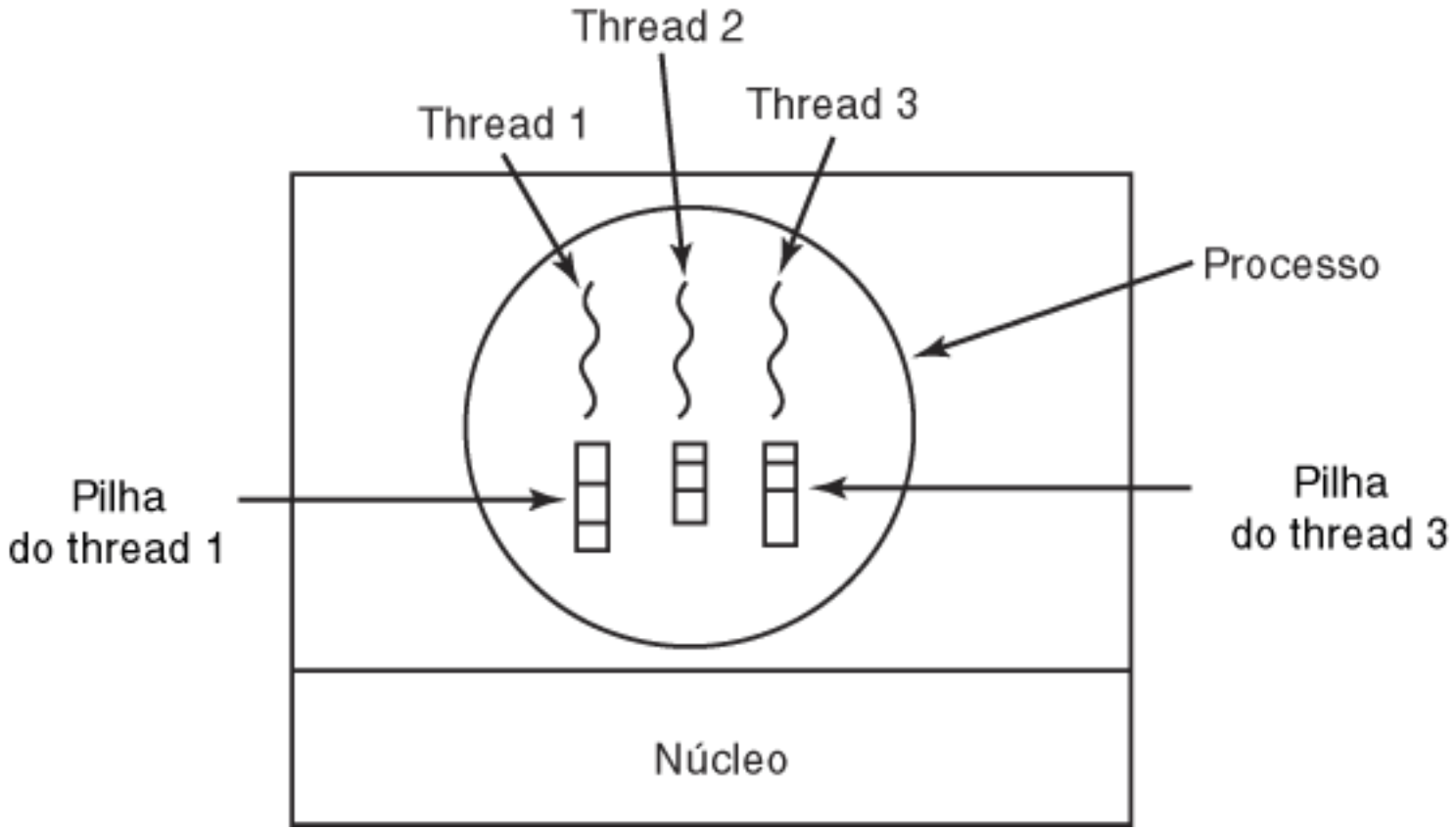
(a) Três processos, cada um com uma thread

(b) Um processo com três threads

Thread


Itens por processo
Espaço de endereçamento
Variáveis globais
Arquivos abertos
Processos filhos
Alarmes pendentes
Sinais e tratadores de sinais
Informação de contabilidade
Itens por thread
Contador de programa
Registradores
Pilha
Estado
compartilhados
privados

Thread



Cada thread tem sua própria pilha

Threads

- Porque?
  - Programas → múltiplas atividades
    - Com vários processos poderiam ser inviável (ou bastante complexo)
  - Criar Thread é mais barato que criar processos
    - Não tem recursos associados
  - Paralelismo...

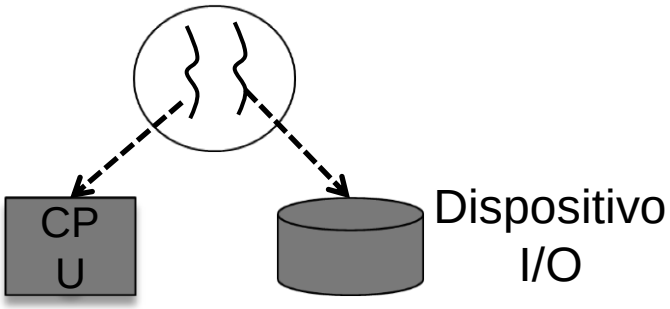
Razões para ter processos leves (*threads*)

- Em muitas aplicações, várias atividades acontecem ao mesmo tempo – **mundo real**
  - O **modelo de programação** (*modelando o mundo real*) se torna mais simples (ou *realista*) decompondo uma aplicação em várias *threads* sequenciais que executam em "paralelo"

- Dado que *threads* são mais leves do que processos, elas são mais fácies (rápidas) de criar e destruir do que processos
  - Criar uma *thread* pode ser **10-100 vezes mais rápido** que criar um processo
  - Quando a necessidade do número de *threads* muda dinâmica e rapidamente, esta propriedade é bastante útil

Razões para ter *threads* (cont)

- Ganhos de velocidade em processos onde atividades de I/O e de computação (CPU) podem ser sobrepostas



  - ***Threads* não levam a ganhos de desempenho quando todas são *CPU-bound***

- *Threads* são úteis, claro, em sistemas com múltiplas CPUs ou *cores* ➔ paralelismo real

Threads

- Comunicação entre processos sem threads: como fazer?
  - Memória compartilhada

Memória Compartilhada - Servidor

```
key = 5678;

/*
 * Create the segment.
 */
if ((shmid = shmget(key, SHMSZ, IPC_CREAT | 0666)) < 0) {
    perror("shmget");
    exit(1);
}

/*
 * Now we attach the segment to our data space.
 */
if ((shm = shmat(shmid, NULL, 0)) == (char *) -1) {
    perror("shmat");
    exit(1);
}

/*
 * Now put some things into the memory for the
 * other process to read.
 */
s = shm;

for (c = 'a'; c <= 'z'; c++)
    *s++ = c;
*s = NULL;

/*
 * Finally, we wait until the other process
 * changes the first character of our memory
 * to '*', indicating that it has read what
 * we put there.
 */
while (*shm != '*')
    sleep(1);
```

Escrevendo dados na memória compartilhada...

Esperando um “sinal” (caractere mudar)

Threads

- Comunicação entre processos sem threads: como fazer?
  - Memória compartilhada
- Outras opções:
  - Arquivo
  - Socket/localhost

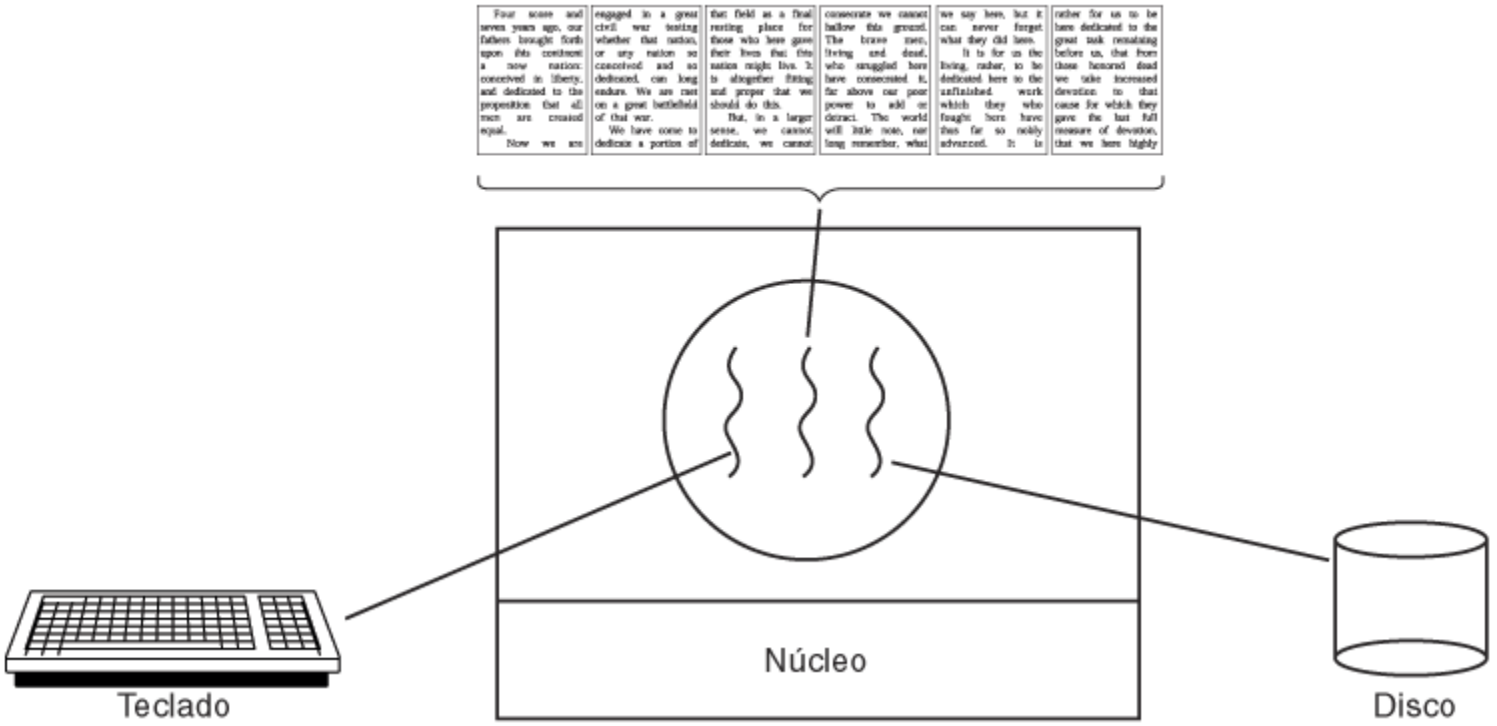
Memória Compartilhada - Cliente

```
/*
 * We need to get the segment named
 * "5678", created by the server.
 */
key = 5678;

/*
 * Locate the segment.
 */
if ((shmid = shmget(key, SHMSZ, 0666)) < 0) {
    perror("shmget");
    exit(1);
}

/*
 * Now we attach the segment to our data space.
 */
if ((shm = shmat(shmid, NULL, 0)) == (char *) -1) {
    perror("shmat");
    exit(1);
}

/*
 * Now read what the server put in the memory.
 */
for (s = shm; *s != NULL; s++)
    putchar(*s);
putchar('\n');

/*
 * Finally, change the first character of the
 * segment to '*', indicating we have read
 * the segment.
 */
*shm = '*';
```

Coloca em stdout (tela) dados da mem. compartilhada

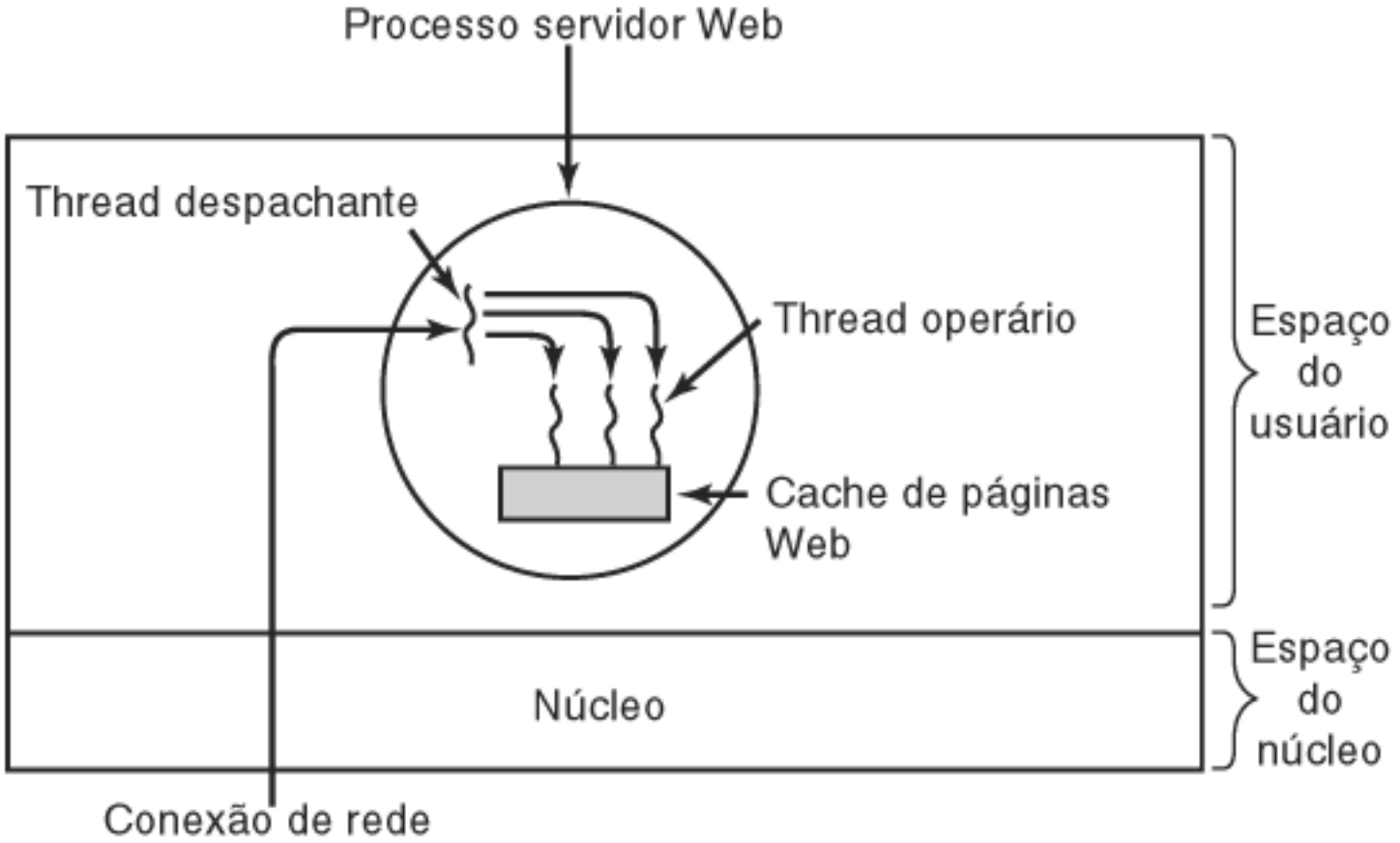
Muda o primeiro caractere – simulando um “sinal”

Uso de Thread – Exemplo 1



Um processador de texto com três threads

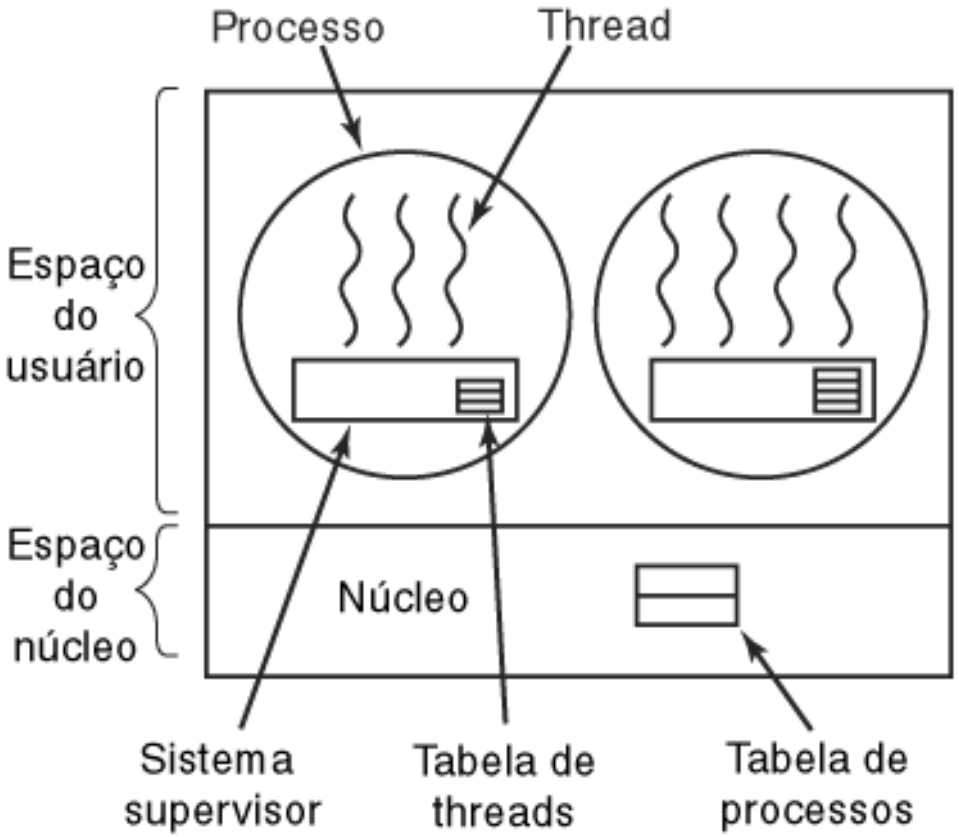
Uso de Thread – Exemplo 2



Um servidor web com múltiplas threads

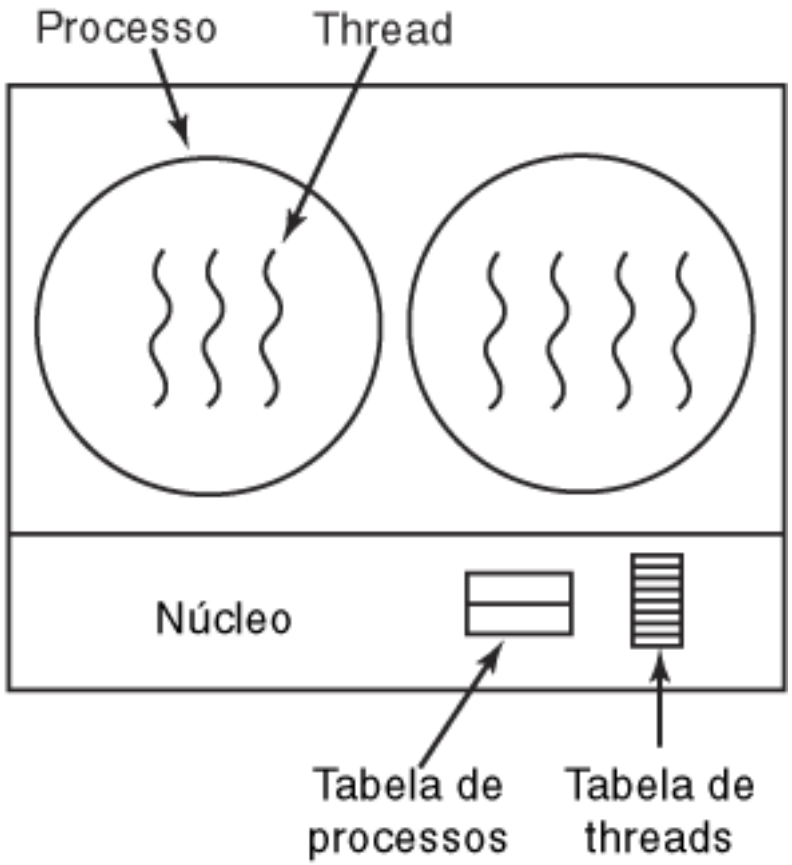
vs um serviço Web com múltiplos servidores (mais adiante – módulo II)

Implementação de Threads de Usuário



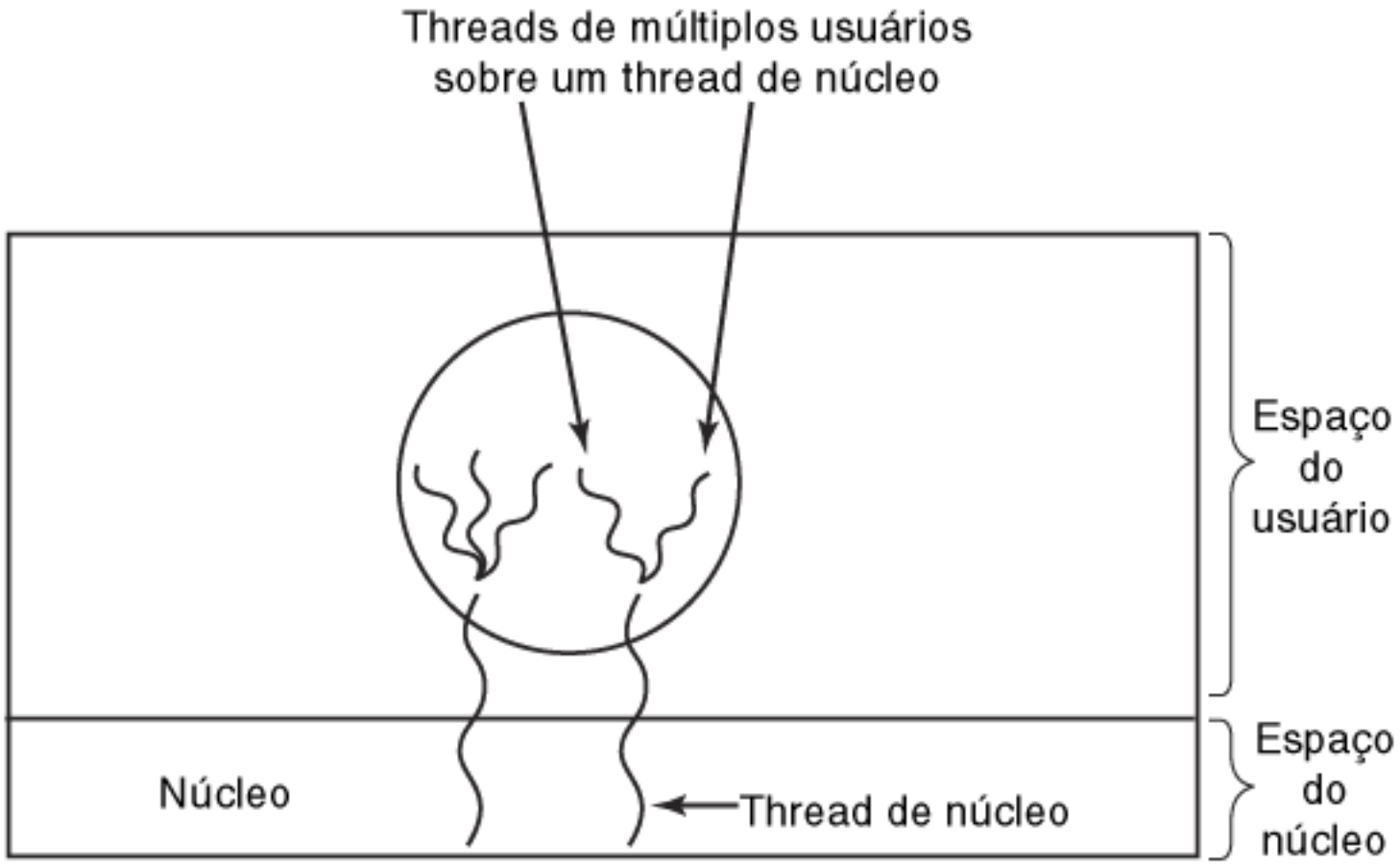
Um pacote de threads de usuário

Implementação de Threads de Núcleo



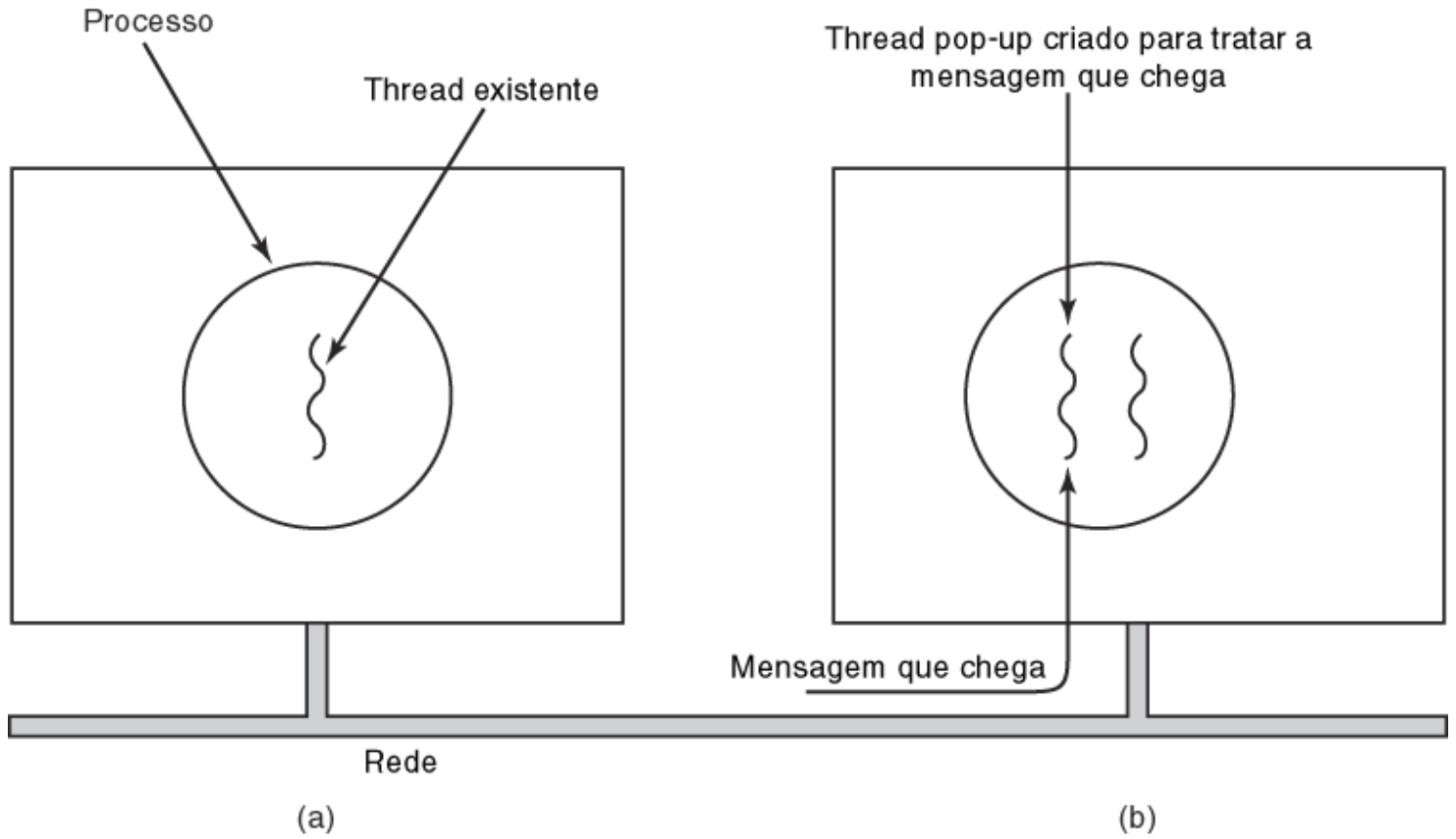
Um pacote de threads gerenciado pelo núcleo

Implementações Híbridas



Multiplexação de threads de usuário sobre threads de núcleo

Criação de um novo thread quando chega uma mensagem



POSIX Threads (1)

- Padrão IEEE POSIX 1003.1c (1995)
  - Portable Operating System Interface
- Pthreads são um conjunto de bibliotecas para a linguagem C, que podem ser implementadas como uma biblioteca a parte ou parte da própria biblioteca C.
- Existem versões da biblioteca
- Cerca de 60 subrotinas
- Algumas chamadas de funções Pthreads:

Thread call	Description
Pthread_create	Create a new thread
Pthread_exit	Terminate the calling thread
Pthread_join	Wait for a specific thread to exit
Pthread_yield	Release the CPU to let another thread run
Pthread_attr_init	Create and initialize a thread's attribute structure
Pthread_attr_destroy	Remove a thread's attribute structure

Comparação de desempenho: processo x thread

Plataforma	fork()	pthread_create()
AMD 2.4 GHz Opteron (8 cpus/node)	41.07	0.66
IBM 1.9 GHz POWER5 p5575 (8 cpus/node)	64.24	1.75
IBM 1.5 GHz POWER4 (8 cpus/node)	104.05	2.01
INTEL 2.4 GHz Xeon (2 cpus/node)	54.95	1.64
INTEL 1.4 GHz Itanium2 (4 cpus/node)	54.54	2.03

Tempos em ms

POSIX Threads - Code

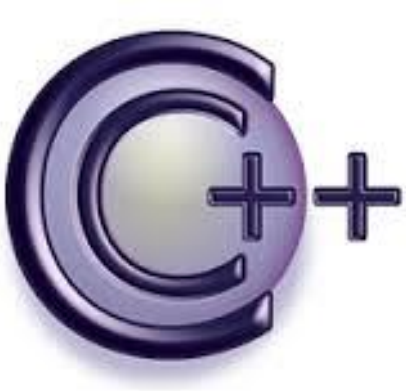
```
#include <pthread.h>
#include <stdio.h>
#include <stdlib.h>

#define NUMBER_OF_THREADS     10

void *print_hello_world(void *tid)
{
        /* This function prints the thread's identifier and then exits. */
        printf("Hello World. Greetings from thread %d0, tid);
        pthread_exit(NULL);
}

        . . .
```

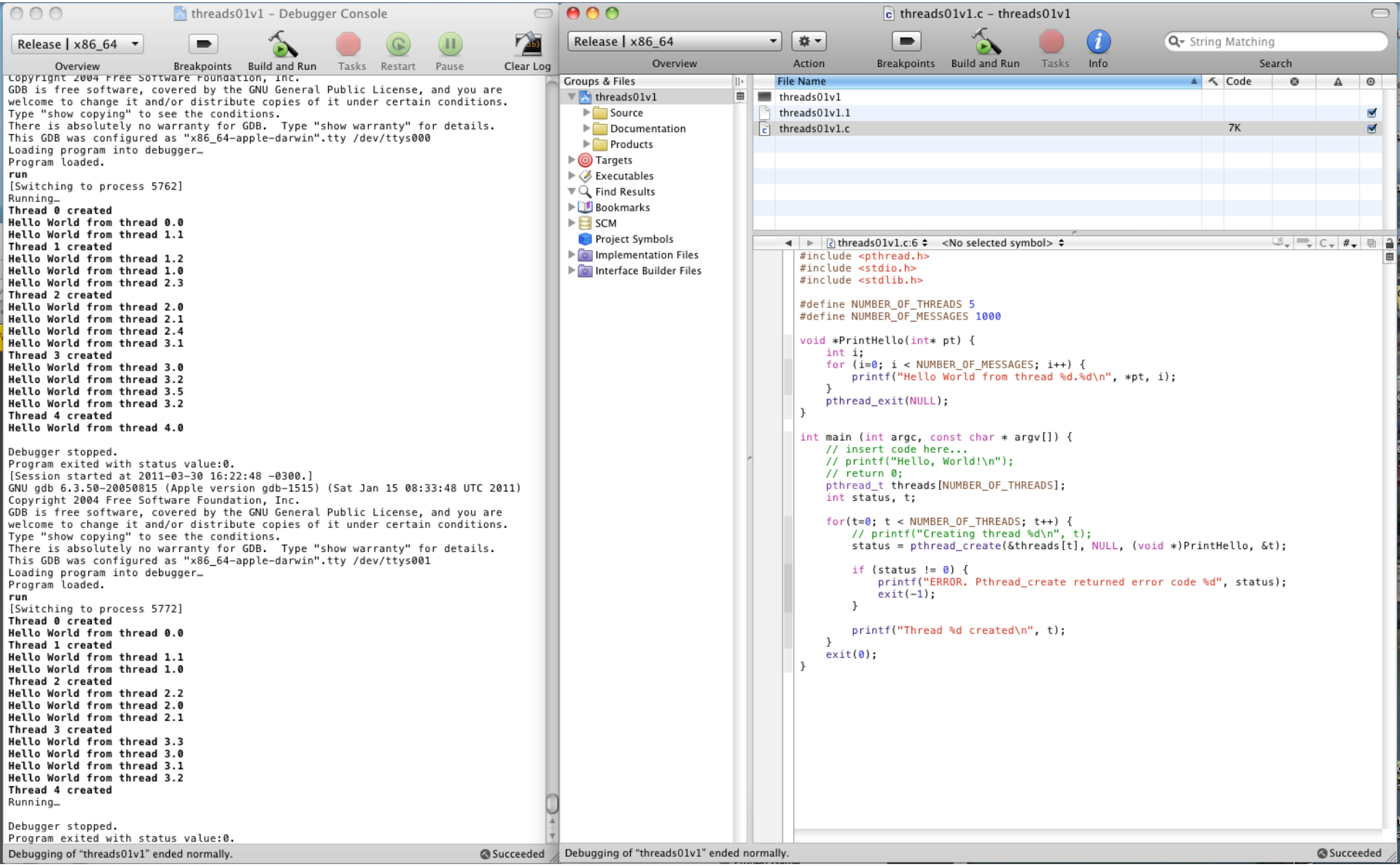
POSIX Threads - Code

```
#include <pthread.h>
#include <stdio.h>
#include <stdlib.h>

#define NUMBER_OF_THREADS     10

void *print_hello_world(void *tid)
{
      /* This function prints the thread's identifier and then exits. */
      printf("Hello World. Greetings from thread %d0, tid);
      pthread_exit(NULL);
}

      int main(int argc, char *argv[])
      {
            /* The main program creates 10 threads and then exits. */
            pthread_t threads[NUMBER_OF_THREADS];
            int status, i;

            for(i=0; i < NUMBER_OF_THREADS; i++) {
                  printf("Main here. Creating thread %d0, i);
                  status = pthread_create(&threads[i], NULL, print_hello_world, (void *)i);

                  if (status != 0) {
                        printf("Oops. pthread_create returned error code %d0, status);
                        exit(-1);
                  }
            }
            exit(NULL);
      }
```

Exercício

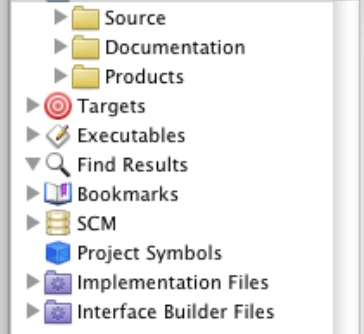
- Executando **n** vezes e verificando a ordem de execução das *threads* criadas …
- Linguagem C




threads01v1 – Debugger Console
Release | x86_64
Overview
Breakpoints
Build and Run
Tasks
Restart
Pause
Clear Log
Copyright 2004 Free Software Foundation, Inc.
GDB is free software, covered by the GNU General Public License, and you are
welcome to change it and/or distribute copies of it under certain conditions.
Type "show copying" to see the conditions.
There is absolutely no warranty for GDB. Type "show warranty" for details.
This GDB was configured as "x86_64-apple-darwin".tty /dev/ttys000
Loading program into debugger…
Program loaded.
run
[Switching to process 5762]
Running…
Thread 0 created
Hello World from thread 0.0
Hello World from thread 1.1
Thread 1 created
Hello World from thread 1.2
Hello World from thread 1.0
Hello World from thread 2.3
Thread 2 created
Hello World from thread 2.0
Hello World from thread 2.1
Hello World from thread 2.4
Hello World from thread 3.1
Thread 3 created
Hello World from thread 3.0
Hello World from thread 3.2
Hello World from thread 3.5
Hello World from thread 3.2
Thread 4 created
Hello World from thread 4.0
Debugger stopped.
Program exited with status value:0.
[Session started at 2011-03-30 16:22:48 -0300.]
GNU gdb 6.3.50-20050815 (Apple version gdb-1515) (Sat Jan 15 08:33:48 UTC 2011)
Copyright 2004 Free Software Foundation, Inc.
GDB is free software, covered by the GNU General Public License, and you are
welcome to change it and/or distribute copies of it under certain conditions.
Type "show copying" to see the conditions.
There is absolutely no warranty for GDB. Type "show warranty" for details.
This GDB was configured as "x86_64-apple-darwin".tty /dev/ttys001
Loading program into debugger…
Program loaded.
run
[Switching to process 5772]
Thread 0 created
Hello World from thread 0.0
Thread 1 created
Hello World from thread 1.1
Hello World from thread 1.0
Thread 2 created
Hello World from thread 2.2
Hello World from thread 2.0
Hello World from thread 2.1
Thread 3 created
Hello World from thread 3.3
Hello World from thread 3.0
Hello World from thread 3.1
Hello World from thread 3.2
Thread 4 created
Running…
Debugger stopped.
Program exited with status value:0.
Debugging of "threads01v1" ended normally.
Succeeded
threads01v1.c – threads01v1
Release | x86_64
Overview
Action
Breakpoints
Build and Run
Tasks
Info
String Matching
Search
Groups & Files
threads01v1
Source
Documentation
Products
Targets
Executables
Find Results
Bookmarks
SCM
Project Symbols
Implementation Files
Interface Builder Files
File Name
Code
threads01v1
threads01v1.1
threads01v1.c
7K
threads01v1.c:6
<No selected symbol>
#include <pthread.h>
#include <stdio.h>
#include <stdlib.h>
#define NUMBER_OF_THREADS 5
#define NUMBER_OF_MESSAGES 1000
void *PrintHello(int* pt) {
int i;
for (i=0; i < NUMBER_OF_MESSAGES; i++) {
printf("Hello World from thread %d.%d\n", *pt, i);
}
pthread_exit(NULL);
}
int main (int argc, const char * argv[]) {
// insert code here...
// printf("Hello, World!\n");
// return 0;
pthread_t threads[NUMBER_OF_THREADS];
int status, t;
for(t=0; t < NUMBER_OF_THREADS; t++) {
// printf("Creating thread %d\n", t);
status = pthread_create(&threads[t], NULL, (void *)PrintHello, &t);
if (status != 0) {
printf("ERROR. Pthread_create returned error code %d", status);
exit(-1);
}
printf("Thread %d created\n", t);
}
exit(0);
}
Debugging of "threads01v1" ended normally.
Succeeded

```
Type "show copying" to see the conditions.
There is absolutely no warranty for GDB.  Type "show warranty" for details.
This GDB was configured as "x86_64-apple-darwin".tty /dev/ttys000
Loading program into debugger…
Program loaded.
run
[Switching to process 5762]
Running…
Thread 0 created
Hello World from thread 0.0
Hello World from thread 1.1
Thread 1 created
Hello World from thread 1.2
Hello World from thread 1.0
Hello World from thread 2.3
Thread 2 created
Hello World from thread 2.0
Hello World from thread 2.1
Hello World from thread 2.4
Hello World from thread 3.1
Thread 3 created
Hello World from thread 3.0
Hello World from thread 3.2
Hello World from thread 3.5
Hello World from thread 3.2
Thread 4 created
Hello World from thread 4.0

Debugger stopped.
Program exited with status value:0.
[Session started at 2011-03-30 16:22:48 -0300.]
GNU gdb 6.3.50-20050815 (Apple version gdb-1515) (Sat Jan 15 08:33:48 UTC 2011)
Copyright 2004 Free Software Foundation, Inc.
GDB is free software, covered by the GNU General Public License, and you are
welcome to change it and/or distribute copies of it under certain conditions.
Type "show copying" to see the conditions.
There is absolutely no warranty for GDB.  Type "show warranty" for details.
This GDB was configured as "x86_64-apple-darwin".tty /dev/ttys001
Loading program into debugger…
Program loaded.
run
[Switching to process 5772]
Thread 0 created
Hello World from thread 0.0
Thread 1 created
Hello World from thread 1.1
Hello World from thread 1.0
Thread 2 created
Hello World from thread 2.2
Hello World from thread 2.0
Hello World from thread 2.1
Thread 3 created
Hello World from thread 3.3
Hello World from thread 3.0
Hello World from thread 3.1
Hello World from thread 3.2
Thread 4 created
Running…

Debugger stopped.
Program exited with status value:0.
```

Debugging of "threads01v1" ended normally. Succeeded



threads01v1.1
threads01v1.c

threads01v1.c:6 <No selected symb

```c
#include <pthread.h>
#include <stdio.h>
#include <stdlib.h>

#define NUMBER_OF_THREADS 5
#define NUMBER_OF_MESSAGES 1000

void *PrintHello(int* pt) {
    int i;
    for (i=0; i < NUMBER_OF_MESSAGES
        printf("Hello World from thr
    }
    pthread_exit(NULL);
}

int main (int argc, const char * arg
    // insert code here...
    // printf("Hello, World!\n");
    // return 0;
    pthread_t threads[NUMBER_OF_THRE
    int status, t;

    for(t=0; t < NUMBER_OF_THREADS;
        // printf("Creating thread %
        status = pthread_create(&thr

        if (status != 0) {
            printf("ERROR. Pthread_c
            exit(-1);
        }

        printf("Thread %d created\n"
    }
    exit(0);
}
```

Debugging of "threads01v1" ended normally.

Exercício

- Threads em Java

Threads: Você no controle

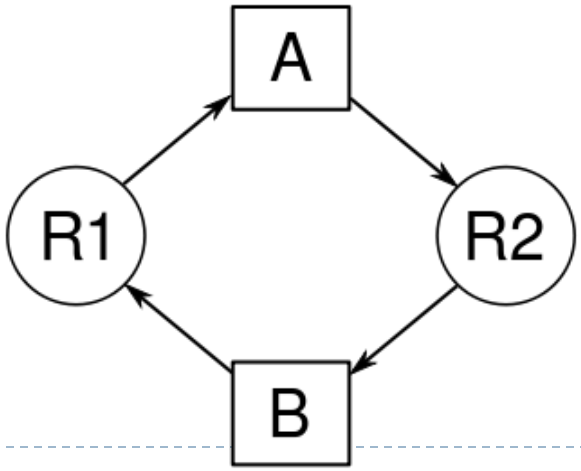
- Processos
  - S.O. gerencia
- Threads
  - Você (programador) tem o controle

Podemos ter problemas?

Conceitos

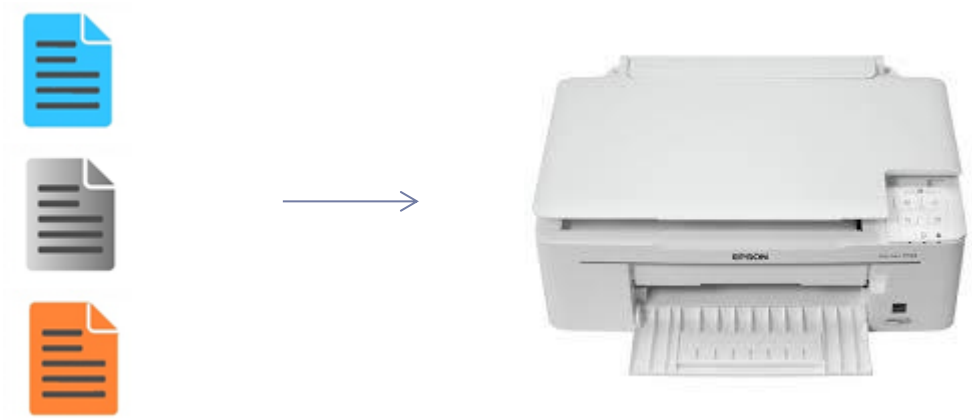
- *Deadlock*
  - Um sistema de bibliotecas só fornece o “nada consta” para alunos matriculados e o sistema de matricula só matricula os alunos perante a apresentação do “nada consta”

  – **Definição**: dois processos bloqueiam a sua execução pois um precisa de um recurso bloqueado pelo outro processo



Conceitos

- Starvation (ou Inanição)
  - Processo espera indefinidamente, e nunca é escolhido para ser executado, “morrendo de fome”.
  - Ex.



Arquivo escolhido sempre o menor...

Solução:

- Aumento de prioridade baseado no tempo de espera
- Primeiro a chegar, primeiro a ser impresso

Conceitos: Concorrência...

Conceitos: **starvation** e **deadlock**

Veremos mais detalhes

Sistemas Operacionais
Processos / Threads

Carlos Ferraz (cagf@cin.ufpe.br)
Jorge Cavalcanti Fonsêca (jcbf@cin.ufpe.br)