

TempOS: Um SO voltado para o ensino de Sistemas Operacionais



Renê de Souza Pinto

Laboratório de Sistemas Distribuídos e
Programação Concorrente
LaSDPC / ICMC - USP
rene@icmc.usp.br
<http://renesp.com.br>

7 de Setembro de 2010

Sumário

Introdução

TempOS

- Multiboot

- Drivers

- Gerenciamento de Memória

- Build System

Próximos Passos

Referências

Motivação

- Curiosidade em desvendar o funcionamento do PC
- Engenharia de computação: O curso abrange desde o mais baixo nível do hardware (construção de transistores) até o mais alto nível de software com a abstração das linguagens de alto nível
- **Sistema Operacional (SO):** Elo entre Hardware e Software
- Falta de uma disciplina prática que abordasse a implementação de um SO

Projeto TempOS

- **2009:** Início do desenvolvimento do Sistema Operacional **TempOS**
- **2010:** **TempOS** torna-se um projeto de mestrado:
 - Elaborar uma plataforma para ensino e treinamento em desenvolvimento de Sistemas Operacionais
 - Material didático, forte documentação do código, exercícios práticos e o próprio TempOS como plataforma de estudo e desenvolvimento

Necessidades

- Grande quantidade de hardware embarcado atual que roda Sistemas Operacionais sofisticados (como Linux)
- Projetos abertos, como o Linux, necessitam de programadores novos - <http://fprudente.blogspot.com/2010/04/por-que-o-linux-nao-esta-atraindo.html>
- Necessidade de mercado: Programadores de *núcleo* estão sendo cada vez mais requisitados (muitas empresas investem, desenvolvem, *customizam* e patrocinam sistemas como o Linux)
- Prover uma ferramenta de estudo factível aos alunos (sistemas muito complexos são difíceis de se estudar)
- Qualificar os alunos dos cursos de computação com tecnologias de ponta

Trabalhos Relacionados

- Nachos[Christopher, Procter e Anderson 1993] - <http://www.cs.washington.edu/homes/tom/nachos/>
- System/161[Holland, Lim e Seltzer 2002] - <http://www.eecs.harvard.edu/~syrah/os161/>
- GeekOS - <http://code.google.com/p/geekos/>
- Minix[Tanenbaum 1987] - <http://www.minix3.org/>
- Outros

Nachos

- Desenvolvido pela Universidade de Berkeley, CA
- O núcleo do Nachos roda em cima de outro Sistema Operacional, simulando uma arquitetura MIPS R2000/3000. Caso esteja executando em uma arquitetura IA-32, é necessário um compilador *cross* para compilar os aplicativos que irão rodar no simulador do Nachos
- Na utilização descrita, os estudantes recebem apenas um esqueleto do código, e durante as práticas vão desenvolvendo as funcionalidades não implementadas

System/161, OS/161

- Desenvolvido pela Universidade de Harvard
- Simula uma arquitetura de hardware (MIPS) podendo-se rodar um Sistema Operacional em cima (OS/161)

GeekOS

- Desenvolvido pela Universidade Marryland, College Park
- Suporta arquitetura IA-32 (x86), podendo rodar tanto no hardware real quanto em emuladores (*bochs*)
- É basicamente um esqueleto de SO, nas disciplinas de laboratório os alunos desenvolvem funcionalidades através de práticas individuais durante o período de um semestre

Minix

- Escrito por Andrew S. Tanenbaum entre 1984 e 1987
- Projeto mais notável na área
- Clássico livro texto “*Operating Systems Design and Implementation*” que aborda o Minix
- Versão 3 do Minix deixou de ter intuito educacional

Outros

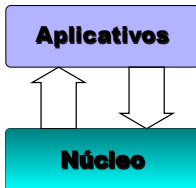
- Xinu[Carissimo 1995]: Desenvolvido na década de 1980 pela Purdue University, originalmente rodava em um microcomputador LSI-11, da Digital Equipment, sendo portado posteriormente para o MS-DOS. Não é mais utilizado.
- Topsy[Fankhauser et al. 1996]: Desenvolvido pelo Swiss Federal Institute of Technology in the Computer Engineering and Networks Laboratory (TIK)
 - Roda em hardware real (arquitetura MIPS R3000) e em emuladores. Possui um manual (70 páginas) escrito em inglês
 - Utilizado em apenas alguns campi europeus[Anderson e Nguyen 2005]

Escrevendo um SO: Um desafio possível

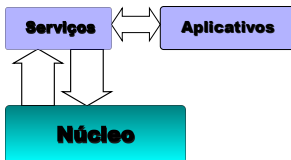
- Escrever um SO necessita de um vasto conhecimento prévio:
 - Organização de computadores
 - Arquitetura de computadores
 - Programação de computadores (C, Assembly)
 - Sistemas Operacionais (teoria)
 - Conhecimento específico de determinados tipos de hardware
- Curiosidade (desvendar mistérios antes obscuros)
- Paciência (seja carinhoso com seu hardware ;)

- Sistema Operacional: Segundo Tanenbaum[Tanenbaum 2000], pode ser visto como um **gerenciador de recursos** ou como uma **máquina estendida**.
- Comunica-se e controla o hardware da máquina, abstraíndo-o através de uma interface (chamadas de sistema)
- Núcleo (*kernel*):
 - Monolítico
 - Micronúcleo
 - Híbrido
 - Exonúcleo

- Monolítico:



- Micronúcleo:



TempOS

- **TempOS:** *TempOS is an educational and multi purpose Operating System*
- Site do projeto: <http://tempos-project.org>
- Escrito exclusivamente em Linguagem C e Assembly (sintaxe AT&T)
- Desenvolvido totalmente no GNU/Linux e com ferramentas livres
- Possui um sofisticado sistema de compilação, pode ser compilado e emulado sem a necessidade de permissões de super-usuário (*root*)
- Roda em um PC comum e em emuladores (qemu)
- Licenciado sob a GNU GPLv2 - *GNU General Public License*, versão 2

TempOS

- Segue a especificação de *Multiboot* (bootável pelo GRUB)
- Funções da Biblioteca C implementadas (strcat, memcpy, printf, etc)
- Tipos de dados padronizados (uint32_t, ulong_t, etc)
- Paginamento de memória permitindo implementação de *swap*
- Realocação dinâmica do Kernel (0-3GB para usuário / 3-4GB para Kernel)
- Sistema de compartilhamento de IRQs
- Drivers para Video (modo texto), Teclado, Temporizador (PIT) e Controlador de Interrupções (PIC)
- Chamadas de sistema (*systemcalls*)
- Port para arquitetura IA-64 (x86 64 bits) iniciado

TempOS - Multiboot

- <http://www.gnu.org/software/grub/manual/multiboot/multiboot.html>
- Especificação criada pela GNU e implementada através do GRUB (GRand Unified Bootloader), permite que vários Sistemas Operacionais “convivam na mesma máquina”, permitindo ao usuário escolher qual iniciar na inicialização da máquina
- “Magic number” deve estar nos primeiros 8192 bytes do núcleo (executável) a ser iniciado
- GRUB facilita a vida dos projetistas de SO: Entra em modo protegido, efetua operações necessárias com a BIOS, etc...
- O TempOS segue a especificação de multiboot, o núcleo do TempOS é compilado em um arquivo ELF que pode ser carregado pelo GRUB

boot.S - Estágio de boot

```

multiboot_header:

.long MULTIBOOT_HEADER_MAGIC
.long MULTIBOOT_HEADER_FLAGS

/* Checksum */
.long -(MULTIBOOT_HEADER_MAGIC + MULTIBOOT_HEADER_FLAGS)

...

/* Initialize the stack pointer */
movl $(stack + STACK_SIZE), %esp

/* Reset EFLAGS */
pushl $0
popf

/* Push multiboot structure and magic value */
pushl %ebx
pushl %ecx

/* Go to kernel, should never return from here */
call EXT_C(karch)

hlt

```

karch.c - Primeiro estágio

```

/**
 * First Stage: Called from Boot Stage.
 */
void karch(unsigned long magic, unsigned long addr)
{
    ...

    /* Init the Memory Manager */
    init_mm();

    /* Setup interrupts */
    setup_IDT();
    init_PIC();
    init_PIT();
    init_IRQ();
    set_picmask(0x00, PIC_MASTER);
    set_picmask(0x00, PIC_SLAVE);
    sti();

    ...

    kprintf(">> First stage done.\n");

    /* Call the TempOS kernel */
    tempos_main(kinf);
}

```

kernel.c - Segundo estágio

```

void tempos_main(karch_t kinf)
{
    /* Start the second stage */
    memcpy(&kinfo, &kinf, sizeof(karch_t));

    /* keep init_timer() in first place because drivers use timer functions */
    init_timer();
    calibrate_delay();

    /* Keyboard */
    init_8042();

    /* ATA controller */
    init_ata_generic();

    /* Init scheduler */
    init_scheduler();

    /* Test */
    kprintf("We are in TempOS kernel!\n");
    kprintf("Command line passed: %s\n", kinfo.cmdline);

    ...
}

```

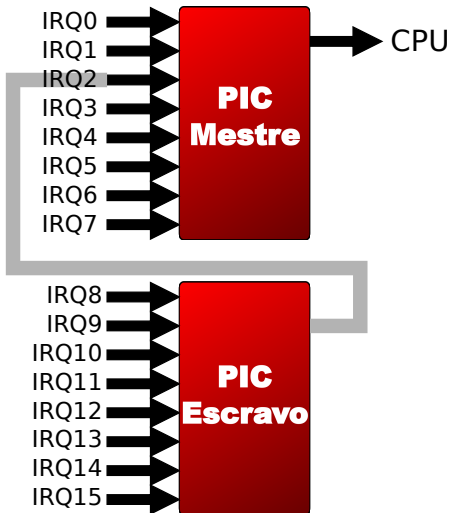
Drivers básicos

- PIT: Temporizador
- PIC: Controlador de Interrupções
- i8024: Controlador de Teclado
- Vídeo: Modo texto
- ATA: Ler/Escriver em HDs PATA (IDE)

Componentes da placa mãe

- PIT - Programmable Interval Timer
 - Conhecido como 8253/8254
 - É um oscilador/divisor de freq. programável
 - Pode ser utilizado para implementar relógio, gerador de ondas, etc
- PIC - Programmable Interrupt Controller
 - É um gerenciador de interrupções
 - Possui 8 entradas
 - Arq. PC: Possui 2 PICs ligados em cascata
 - Controle de até 15 interrupções (1 entrada é para a cascata)
 - Cada entrada é uma IRQ (**I**nterrupt **ReQ**uest)
 - Um PIC atua como Mestre e outro como Escravo:

PIC



PIC - Mestre

- IRQs no Mestre:
 - IRQ 0: Reservada para o PIT (temporizador)
 - IRQ 1: Teclado
 - IRQ 2: Para ligação em cascata com o escravo
 - IRQ 3: Porta serial COM2 (Padrão) ou COM4
 - IRQ 4: Porta serial COM1 (Padrão) ou COM3
 - IRQ 5: Porta paralela LPT2 ou placa de som
 - IRQ 6: Controlador de disquete
 - IRQ 7: Porta paralela LPT1 ou placa de som

PIC - Escravo

- IRQs no Escravo:
 - IRQ 8: RTC (Real Time Clock)
 - IRQ 9: Mapeada para IRQ2 (cascata)
 - IRQ 10: Disponível
 - IRQ 11: Disponível
 - IRQ 12: PS/2 mouse
 - IRQ 13: ISA / Co-processador matemático
 - IRQ 14: Canal IDE primário
 - IRQ 15: Canal IDE secundário

Arquitetura PC

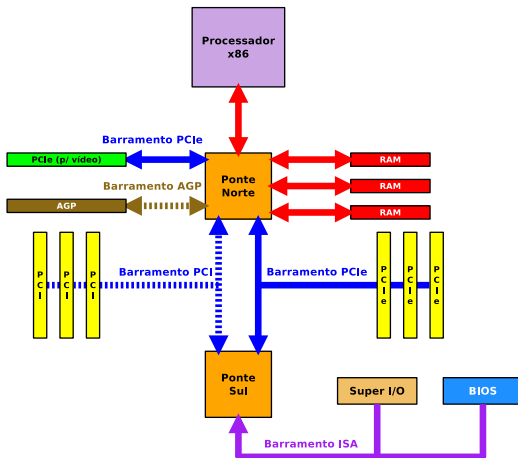


Figura: Arquitetura PC

Componentes da placa mãe

- Ponte Norte:
 - Comunica-se diretamente com o processador e com a Ponte Sul, provê:
 - Controlador de memória (RAM)
 - Controle de barramentos de vídeo (AGP, PCIe x16)
- Ponte Sul:
 - Geralmente implementa controladores para:
 - PCI / PCIe
 - SATA / PATA (antigo IDE)
 - USB
 - Ethernet, Áudio, Modem (onboard)

Controle de interrupções

- Drivers registram sua IRQ:

```
int request_irq(uint16_t irq, void (*handler)(int, pt_regs *),
               uint16_t flags, const char *name)
```

- Na interrupção: Cada rotina registrada para a IRQ em que a interrupção ocorreu é executada, permitindo o compartilhamento de IRQs

Alarmes

- Novos alarmes podem ser criados com:

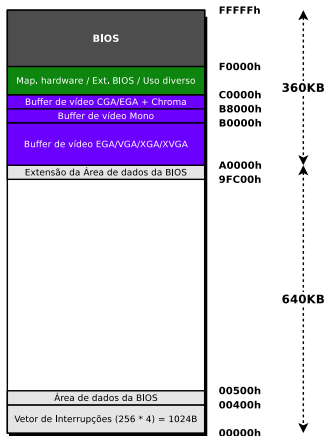
```
int new_alarm(uint32_t expires, void (*handler)(int),
              uint32_t arg)
```

- Exemplo de uso (escalonador do TempOS):

```
void init_scheduler(void)
{
    /* Register alarm to do task switch */
    if(!new_alarm((jiffies + scheduler_quantum), schedule, 0)){
        panic("Could not install scheduler alarm.");
    }
}
```

Driver de Vídeo - Modo Texto

Trabalha diretamente com a memória de vídeo (mapeada para 0xB8000)



Gerenciamento de Memória

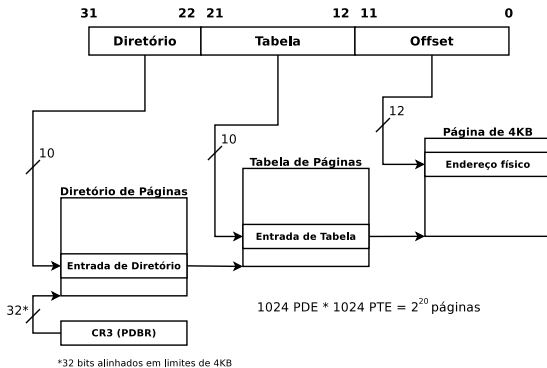
Arquitetura IA-32 (x86):

- Modelos de Memória:
 - **Modelo plano:** A memória é vista como um espaço de endereço contínuo, pode endereçar até $2^{32} - 1$ bytes (4GB)
 - **Modelo Segmentado:** A memória é dividida em segmentos. Até 16383 segmentos são suportados, cada um podendo endereçar até 2^{36} bytes
 - **Modelo Modo de Endereço Real:** Memória dividida em segmentos de 64KB
- Os segmentos são descritos em uma **Tabela Global de Descritores (GDT)**
- Nota: Os segmentos podem se sobrepor!

Arquitetura IA-32 (x86)

■ Proteção de Memória:

- Pode ser feita por segmentação (obsoleto)
- Paginamento (atual):



TempOS - Gerenciamento de Memória

- Alocação de páginas físicas: Pilha de páginas, basta empilhar ou desempilhar
- Alocação do espaço virtual de memória: Feita por mapa de bits. Cada bloco alocado contém um meta dado, indicando posição inicial/final e tamanho do bloco

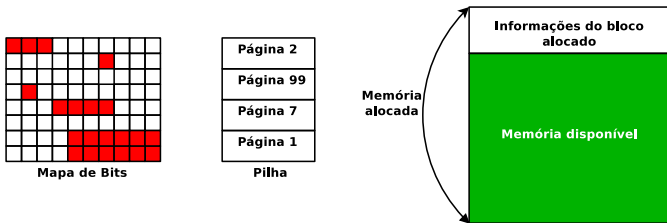


Figura: Estruturas de alocação de memória no TempOS

TempOS - Gerenciamento de Memória

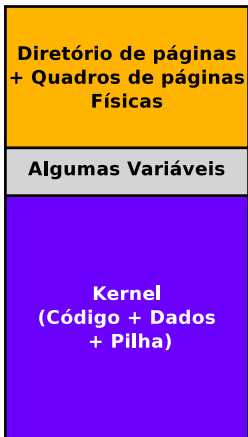


Figura: Imagem do núcleo do TempOS na memória

TempOS - Build System

- Fácil de utilizar:
 - *make*: Compila o núcleo do TempOS
 - *make install*: Gera uma imagem de disco com o TempOS que pode ser iniciada em uma máquina real ou em um emulador
 - *make test*: Testa a imagem gerada com o emulador QEMU

TempOS - Screenshot

```

QEMU - Press Ctrl-Alt to exit grab
TempOS
000000000:00009FC00 - Available
00009FC00:0000A0000 - Reserved
0000E8000:000100000 - Reserved
00054F000:007FF0000 - Available
007FF0000:008000000 - ACPI
000100000:00054F000 - Reserved
>> First stage done.
Initializing timer...
Calibrating delay...44 BogoMIPS
Initializing i8042 keyboard controller...
Initializing generic ATA controller...
 hda: Device not found.
 hdb: Device found
      Model: QEMU HARDDISK
 hdc: CD/DVD-ROM detected.
 hdd: Device not found.
We are in TempOS kernel!
Command line passed: /boot/tempos.elf
1 2 3 74 0 65 6D 0 0 70 6F 0 73 0 0
    
```

Figura: TempOS rodando no QEMU

Próximos Passos

- Unificar a API do TempOS e definir todos os subsistemas do núcleo
- Desenvolver Sistema de Arquivos Virtual e suporte a EXT2 (em progresso)
- Drivers para SATA, chaveamento de processos
- Melhorar o gerenciador de memória
- Desenvolver mais chamadas de sistema
- Portar biblioteca C (newlib) para o TempOS
- Portar gcc para o TempOS e fazer o TempOS compilar a si mesmo!
- Desenvolver o material didático




Para saber mais...

- Projeto TempOS: <http://www.tempos-project.org>
- USP osdev: <http://code.google.com/p/usp-osdev/>
- Bona fide OS developer: <http://www.osdever.net/>
- OS Dev wiki: http://wiki.osdev.org/Main_Page





Dúvidas?

Até Logo!

Referências (1)

- 
 ANDERSON, C. L.; NGUYEN, M. A survey of contemporary instructional operating systems for use in undergraduate courses. *J. Comput. Small Coll.*, Consortium for Computing Sciences in Colleges, , USA, v. 21, n. 1, p. 183–190, 2005. ISSN 1937-4771.
- 
 CARISSIMO, J. XINU—an easy to use prototype for an operating system course. *ACM SIGCSE Bulletin*, ACM, v. 27, n. 4, p. 56, 1995.
- 
 CHRISTOPHER, W.; PROCTER, S.; ANDERSON, T. The Nachos instructional operating system. In: USENIX ASSOCIATION. *Proceedings of the USENIX Winter 1993 Conference Proceedings on USENIX Winter 1993 Conference Proceedings*. [S.l.], 1993. p. 4.

Referências (2)

-  FANKHAUSER, G. et al. Topsy—a teachable operating system. *Computer Engineering and Networks Laboratory, ETH Zurich*, v. 2001, 1996.
-  HOLLAND, D.; LIM, A.; SELTZER, M. A new instructional operating system. In: ACM. *Proceedings of the 33rd SIGCSE technical symposium on Computer science education*. [S.I.], 2002. p. 111–115.
-  TANENBAUM, A. A UNIX clone with source code for operating systems courses. *ACM SIGOPS Operating Systems Review*, ACM, v. 21, n. 1, p. 29, 1987.
-  TANENBAUM, A. S. *Sistemas Operacionais: projeto e implementação*. [S.I.]: Bookman, 2000.