Sistemas Operacionais

Princípios Básicos de Software de E/S

André Luis Martinotto

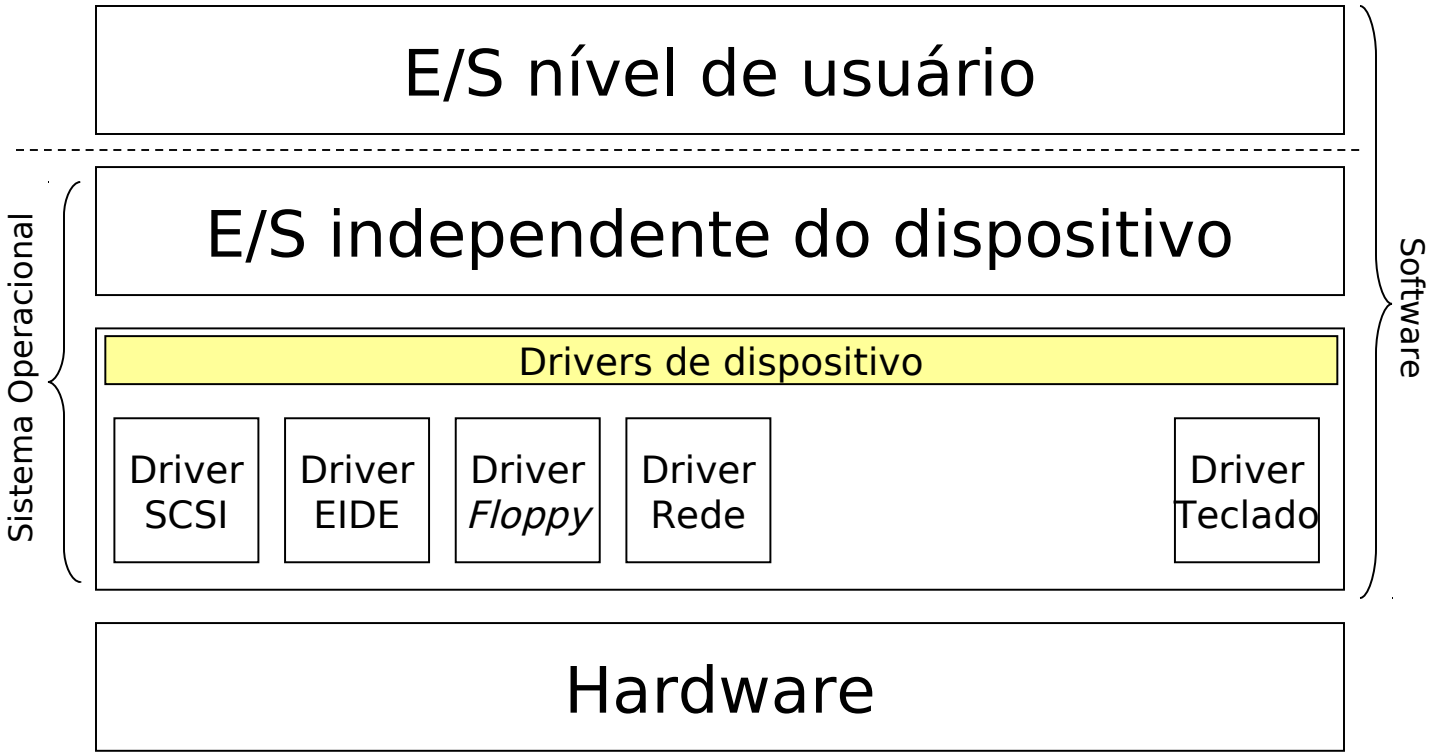
UCS UNIVERSIDADE DE CAXIAS DO SUL

Sistema de Entrada e Saída

- Subsistema de entrada e saída é software complexo devido a diversidade de periféricos
  - Objetivo é **padronizar** as rotinas de acesso aos periféricos de E/S

- Para atingir esse objetivo o subsistema de E/S é organizado em **camadas**
  - Permite inclusão de novos dispositivos sem alterar interface de utilização
  - As camadas mais baixas escondem as peculiaridades do hardware

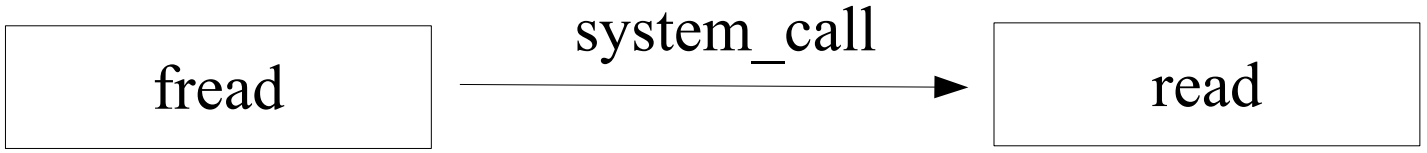
Sistema de Entrada e Saída



Sistema de Entrada e Saída

- **E/S nível de usuário:**
  - Rotinas de I/O executadas por programas através de bibliotecas
  - Normalmente essas rotinas efetuam chamadas de sistema
  - Permite que as rotinas sejam independentes do sistema operacional

bytes_lidos = fread(buffer, tam_item, n_itens, arquivo)



E/S Independente do dispositivo

- Implementa procedimentos e funções gerais a todos os dispositivos de entrada e saída

- **Principais serviços**
  - Escalonamento de E/S
  - Denominação
  - Bufferização
  - Cache de dados
  - Alocação e liberação
  - Direitos de acesso
  - Tratamento de erros

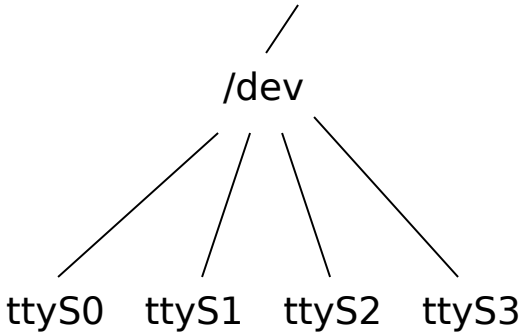
E/S Independente do dispositivo

- **Objetivo:**
  - Fornecer uma visão lógica do dispositivo através de estruturas de dados genéricas que representam classes de dispositivos
  - **Exemplos:** dispositivos de caracteres e bloco

- Atribuição uniforme do nome independente do dispositivo

E/S Independente do dispositivo

- O UNIX é um exemplo clássico:
  - Nome do dispositivo é uma string
  - Dispositivos fazem parte do sistema de arquivos




```
crw------- 1 root tty         4.64 may 5 2000 /dev/ttyS0
crw------- 1 root tty         4.65 may 5 2000 /dev/ttyS1
crw------- 1 root tty         4.66 may 5 2000 /dev/ttyS2
crw------- 1 root tty         4.67 may 5 2000 /dev/ttyS3
```

- **Exemplo:**

cat arquivo.txt > /dev/tty

E/S Independente do dispositivo

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

int main() {
    FILE *fp;
    fp = fopen("/dev/tty", "rw+");

    if ( fp == NULL ){
            perror("Nao foi possivel abrir o terminal\n");
            exit(0);
    }
    fprintf(fp, "Alo mundo\n");
    fclose(fp);
}
```

Driver do Dispositivo

- Todo o código dependente do dispositivo aparece no driver do dispositivo.
  - Cada driver manipula um dispositivo ou uma classe de dispositivos intimamente relacionados
  - Devido a dependência entre os drivers e as chamadas de sistema os fabricantes desenvolvem drivers específicos para cada SO

- **Driver** ⇒ deve ser acoplado ao kernel do. SO

Módulos do Kernel

- Drivers podem ser implementados utilizando módulos do kernel

- **Comandos:**
  - lsmod: lista os módulos carregados
  - insmod: carrega um módulo no kernel
  - rmmod: remove um módulo carregado
  - modprobe: carrega um módulo e suas dependências
  - depmod: checa as dependências dos módulos
  - modinfo: exibe informações sobre um módulo (parâmetros que um módulo aceita, dependências, descrição, etc)

Módulos do Kernel

- Módulos não podem usar bibliotecas do C (printf), mas possuem a disposição funções do núcleo do kernel (printk)
- O Kernel possui um única pilha de 4Kb usada por todos os módulos
  - O programador deve minimizar o número de variáveis locais

Exemplo Módulo - hello.c

```
#include <linux/module.h>
#include <linux/kernel.h>

MODULE_LICENSE("GPL");

int init_module(void) {
    printk ("Hello World\n");
    return 0;
}

void cleanup_module (void) {
    printk ("Goodbye World\n");
}
```

Exemplo - Makefile

```
obj-m       :=      hello.o

KDIR        := /lib/modules/$(shell uname -r)/build
PWD         := $(shell pwd)

Default:
        make -C $(KDIR) SUBDIRS=$(PWD) modules
```

Carregando o Módulo

- **No terminal:**
  - tail -f /var/log/syslog
- **Em outro terminal:**
  - insmod hello.ko
  - rmmod hello.ko

Módulos - Diretivas

- O código fonte pode conter as seguintes diretivas:
  - MODULE_DESCRIPTION(“...”)
  - MODULE_VERSION(“...”)
  - MODULE_ALIAS(“...”)
  - MODULE_LICENSE(“..”)

“GPL”
“GPL v2”
“GPL and additional rights”
“DUAL BSD/GPL”
“DUAL MIT/GPL”
“DUAL MPL/GPL”
“Proprietary”

Módulos - Parâmetros

- Módulos não possuem main()
- Parâmetros são declarados pela diretiva:

  module_param(var,type,access)

- **Tipos podem ser**:
  - bool, invbool (valor booleano invertido), charp (ponteiro para string), int, uint (inteiro sem sinal), short, ushort, long, ulong
- **Os acessos podem ser:**
  - S_IRUGO (apenas leitura), S_IRUGO | S_IWUSR (leitura e escrita).

Exemplo Módulo - contador.c

```
#include <linux/module.h>
#include <linux/kernel.h>
#include <linux/moduleparam.h>

static int valor = 0;
module_param(valor, int, S_IRUGO | S_IWUSR);

MODULE_DESCRIPTION("SOII");
MODULE_LICENSE("GPL");

int init_module(void) {
      int i;
      for (i=0; i<valor; i++){
            printk ("Hello World = %d!\n",i);
      }
      return 0;
}

void cleanup_module (void) {
      printk ("Goodbye World\n");
}
```

Exemplo - Makefile

```
obj-m       :=      contador.o

KDIR        := /lib/modules/$(shell uname -r)/build
PWD         := $(shell pwd)

Default:
        make -C $(KDIR) SUBDIRS=$(PWD) modules
```

Carregando o Módulo

- **No terminal:**
  - tail -f /var/log/syslog

- **Em outro terminal:**
  - insmod contador.ko valor=4
  - modinfo contador.ko
  - rmmod hello.ko