Introdução ao Kernel Linux

Pontifícia Universidade Católica de Campinas

Prof. Dr. Denis M. L. Martins

Disclaimer

Parte do material apresentado a seguir foi adaptado de:

- Linux kernel and driver development training
- Linux Device Drivers, Third Edition

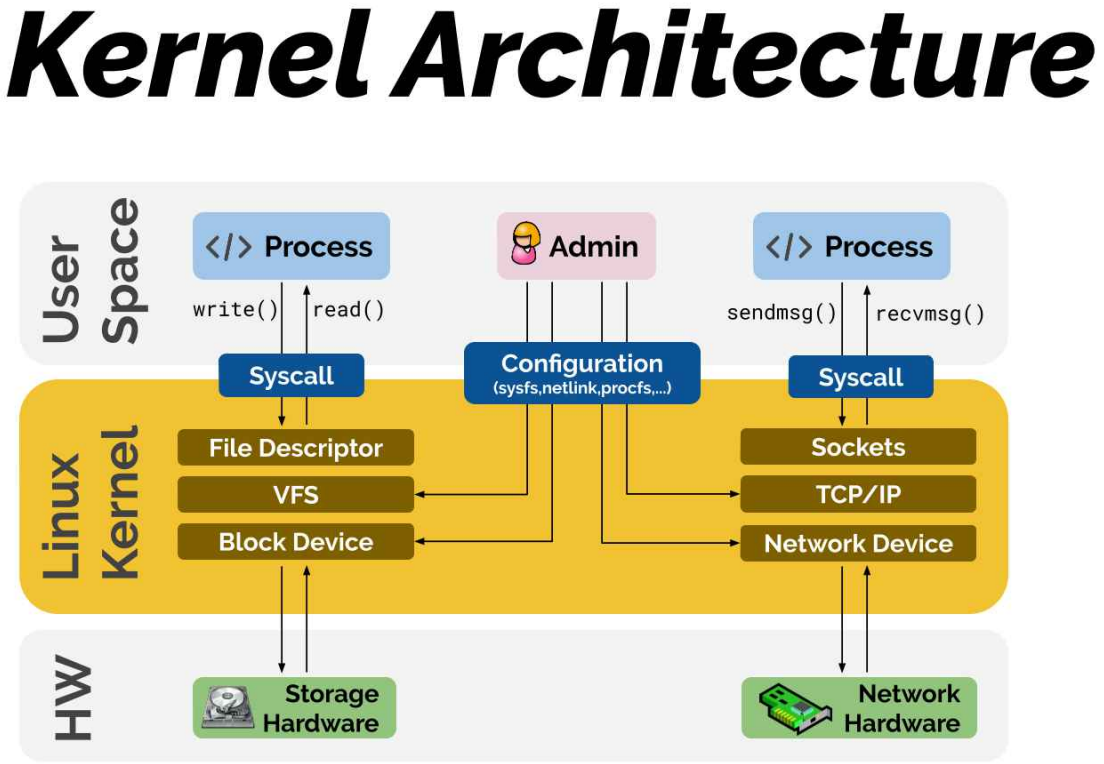
Imagens decorativas (à esquerda dos slides) retiradas de Unsplash

Objetivos de Aprendizagem

- Compreender a estrutura geral do Kernel Linux
- Explicar o conceito de chamadas de sistema
- Compreender o papel dos módulos do Kernel
- Explicar a diferença entre um módulo do kernel e uma aplicação tradicional

Funções do Kernel

- Gerenciar todos os recursos de hardware: CPU, memória e E/S (I/O).
- Fornecer um conjunto de APIs portáveis, independentes da arquitetura e do hardware, para permitir que aplicações e bibliotecas no espaço do usuário utilizem os recursos de hardware.
- Lidar com acessos e uso concorrentes dos recursos de hardware por diferentes aplicações.
- Na imagem: Arquitetura geral do Kernel Linux. Fonte: eBPF - Rethinking the Linux Kernel

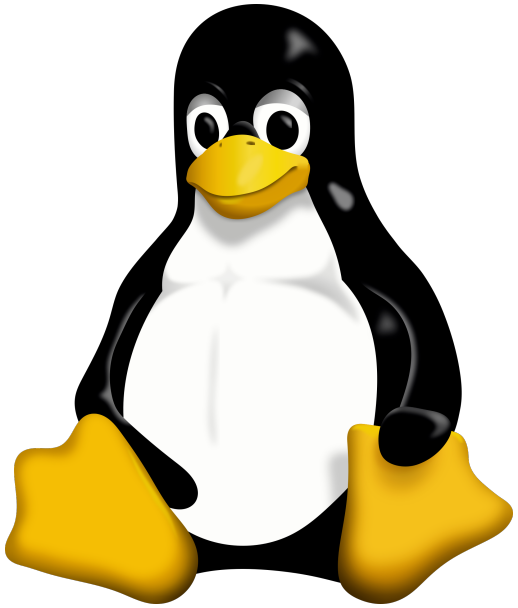


Kernel Linux

- O kernel Linux foi criado como um hobby em 1991 por um estudante finlandês, **Linus Torvalds** (na imagem ao lado - fonte da imagem: Wikipedia).
  - Rapidamente começou a ser utilizado como o kernel para sistemas operacionais de software livre.
- Linus Torvalds conseguiu criar uma grande e dinâmica comunidade de desenvolvedores e usuários em torno do Linux.
  - Atualmente, cerca de 2.000+ pessoas contribuem para cada lançamento do kernel, indivíduos ou empresas grandes e pequenas.
- Linus Torvalds também criou o Git

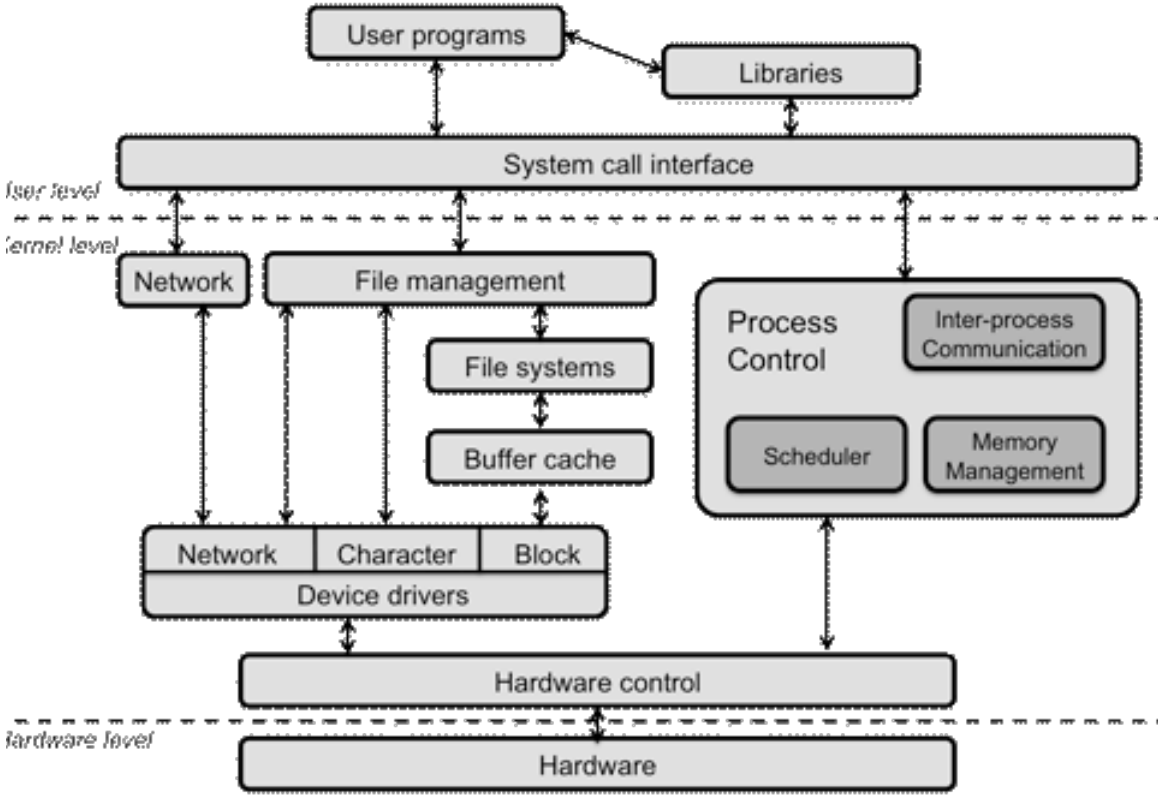
Kernel Linux

- **Implementado em C:** Assim como todos os sistemas UNIX, o kernel é implementado em C.
- **Uso de Assembly:** Uma pequena quantidade de Assembly é usada para:
  - Inicialização da CPU e da máquina.
  - Exceções.
  - Rotinas de biblioteca críticas.
- **Sem C++:** O kernel não usa C++.
- **Compilado com GCC:** Todo o código é compilado com o gcc.
  - Usa muitas extensões específicas do gcc
  - Subconjunto suporta LLVM (Clang)
- O suporte a Rust: `drivers/net/phy/ax88796b_rust.rs`



Kernel Subsystems

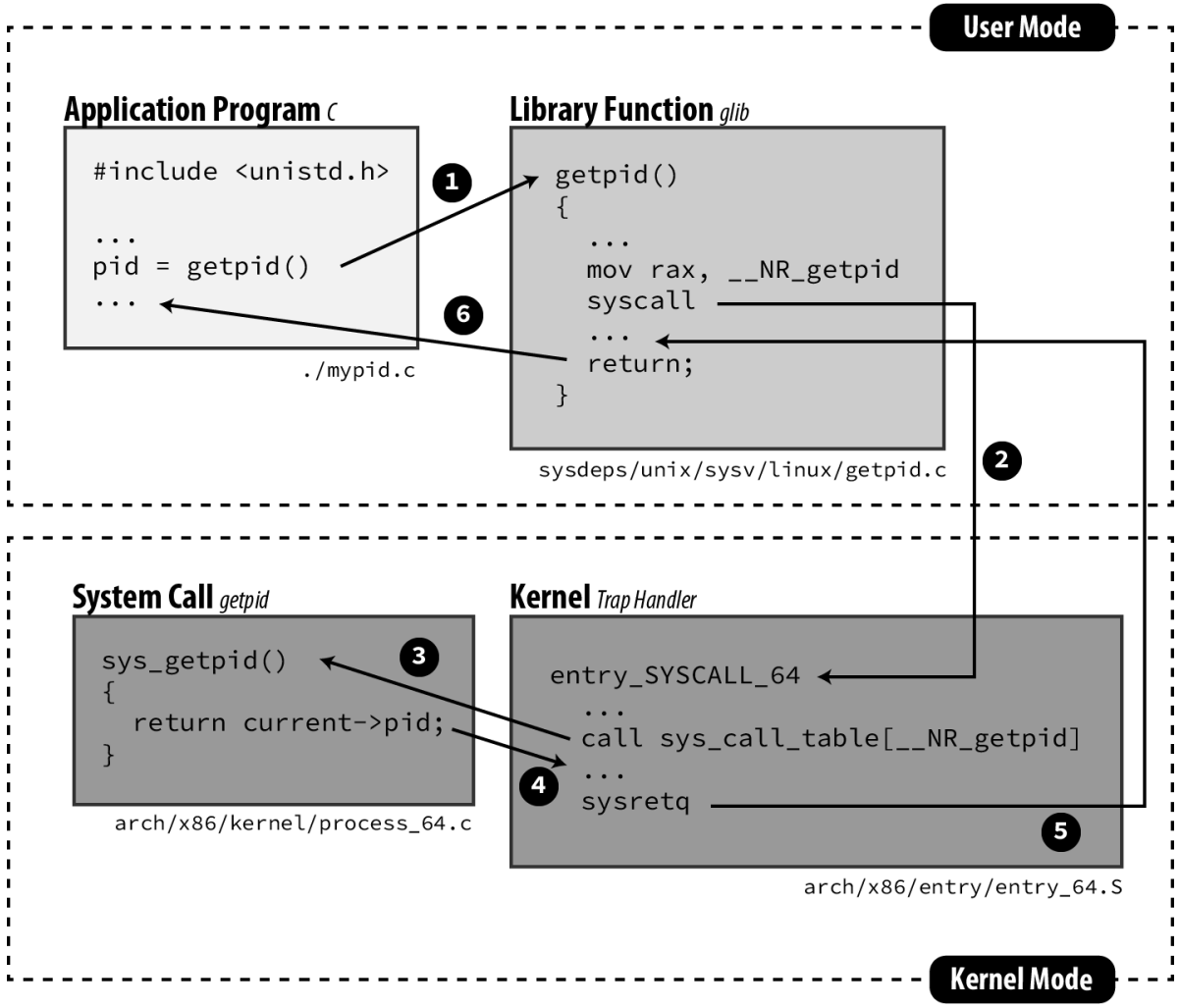
- **System Call Interface:** Interface entre o espaço do usuário e o kernel, permitindo que programas executem funções do sistema operacional.
- **Process Management:** Criação, agendamento e destruição de processos; comunicação entre processos.
- **Memory Management:** Alocação e desalocação de memória física e virtual; paginação, segmentação.
- **Virtual File System:** Abstração para diferentes sistemas de arquivos (ext4, XFS, NTFS, etc.), permitindo que o kernel interaja com eles de forma uniforme.
- **Device Drivers:** Interface entre o kernel e os dispositivos de hardware (discos, placas de rede, impressoras, etc.).
- **Networking:** Implementação dos protocolos TCP/IP; gerenciamento de conexões de rede.



Fonte da imagem: Operating Systems @ Rutgers

System Calls

- A interface entre o sistema operacional e os programas de usuário é o conjunto de chamadas de sistema (`open`, `close`, `read`, `fork`, `execve`, etc.).
- Encapsulada pela biblioteca C: as aplicações geralmente não fazem uma chamada de sistema diretamente, mas sim utilizam a função correspondente da biblioteca C.
- Utiliza o mecanismo de interrupção, executando em modo kernel.
- Exemplo: chamada de sistema `getpid` em sistemas Intel/Linux coloca o número 20 no registrador `eax` (20 é o número correspondente à chamada de sistema `getpid`) e então executa `INT 0x80`, que gera uma interrupção.



Fonte da imagem: Julien Sobczak

`/proc` e `/sys`

- O kernel exporta uma grande quantidade de estatísticas, dados de configuração, etc., através deste sistema de *pseudo-arquivos*.
- Esses "arquivos" não são armazenados em disco em lugar algum: eles são criados e atualizados dinamicamente pelo kernel.
- Os dois sistemas de arquivos pseudo mais importantes são:
  - **proc**, geralmente montado em `/proc`:
    Informações relacionadas ao sistema operacional (processos, parâmetros de gerenciamento de memória...).
  - **sysfs**, geralmente montado em `/sys`: Representação do sistema como uma árvore de dispositivos conectados por barramentos. Informações coletadas pelos frameworks do kernel que gerenciam esses dispositivos.

Módulos de Kernel

- É um fragmento de código que pode ser carregado e descarregado dinamicamente no kernel em tempo de execução.
- Tipicamente escritos em C ou C++ e compilados diretamente para o espaço de endereço do kernel.
- Abordagem flexível no design de Sistemas Operacionais.
- Permite que funcionalidades estam adicionadas sem a necessidade de recompilar todo o kernel. No Linux, use `lsmod` para listar todos os módulos carregados no sistema:

```
$ lsmod
Module                  Size  Used by
tls                   110592  0
binfmt_misc            24576  1
intel_rapl_msr         20480  0
intel_rapl_common      40960  1 intel_rapl_msr
snd_hda_codec_generic  102400  1
ledtrig_audio          16384  1 snd_hda_codec_generic
kvm_intel             421888  0
```

Módulos de Kernel: Exemplo

```c
#include <linux/module.h>    // Necessário para todos os módulos
#include <linux/kernel.h>    // Necessário para KERN_INFO
#include <linux/init.h>      // Necessário para as macros

MODULE_LICENSE("GPL");  // Obrigatório para código licenciado GPL
MODULE_AUTHOR("Seu Nome Aqui");
MODULE_DESCRIPTION("Um módulo de kernel 'Olá, Mundo' simples.");

static int __init hello_init(void) {
    printk(KERN_INFO "Olá, mundo!\n"); // Imprime no log do kernel
    return 0;  // Sucesso
}

static void __exit hello_exit(void) {
    printk(KERN_INFO "Adeus, mundo cruel!\n"); // Imprime ao descarregar o módulo
}

module_init(hello_init);
module_exit(hello_exit);
```

Módulos de Kernel

- Cada módulo pode ser dinamicamente vinculado ao kernel em execução pelo programa `insmod` e desvinculado pelo programa `rmmod`.
- **Função Principal:** Módulos se registram para atender futuras requisições, ao contrário de aplicações tradicionais que executam uma tarefa completa.
- **Inicialização (module_init):** Prepara o módulo para uso futuro; não executa a tarefa em si.
- **Saída (module_exit):** Limpa tudo configurado na inicialização antes do descarregamento, evitando "lixo" no sistema.
- **Programação Orientada a Eventos:** Módulos operam como programas orientados a eventos, reagindo a requisições.
- **Vinculação Limitada:** Módulos só podem chamar funções exportadas pelo kernel, não bibliotecas externas como libc.

Módulos de Kernel

- As linhas `module_init` e `module_exit` usam macros especiais do kernel para indicar o papel dessas duas funções.
- Outra macro especial (`MODULE_LICENSE`) é usada para informar ao kernel que este módulo possui uma licença livre; sem essa declaração, o kernel reclama quando o módulo é carregado.
- A função `printk` é definida no kernel Linux e disponibilizada aos módulos; ela se comporta de forma semelhante à função padrão da biblioteca C `printf`.
    - O kernel precisa de sua própria função de impressão porque ele executa sozinho, sem a ajuda da biblioteca C.
    - O módulo pode chamar `printk` porque, depois que o `insmod` o carrega, o módulo é vinculado ao kernel e pode acessar os símbolos públicos do kernel (funções e variáveis, conforme detalhado na próxima seção).

Conclusão

Revisão dos Pontos Chave:

- Módulos do kernel como extensões dinâmicas e flexíveis.
- A capacidade de descarregar módulos facilita o desenvolvimento e teste iterativos.
- A importância de `module_init` e `module_exit` para registro e limpeza.
- As limitações de recursos (memória, ponto flutuante) no ambiente do kernel.

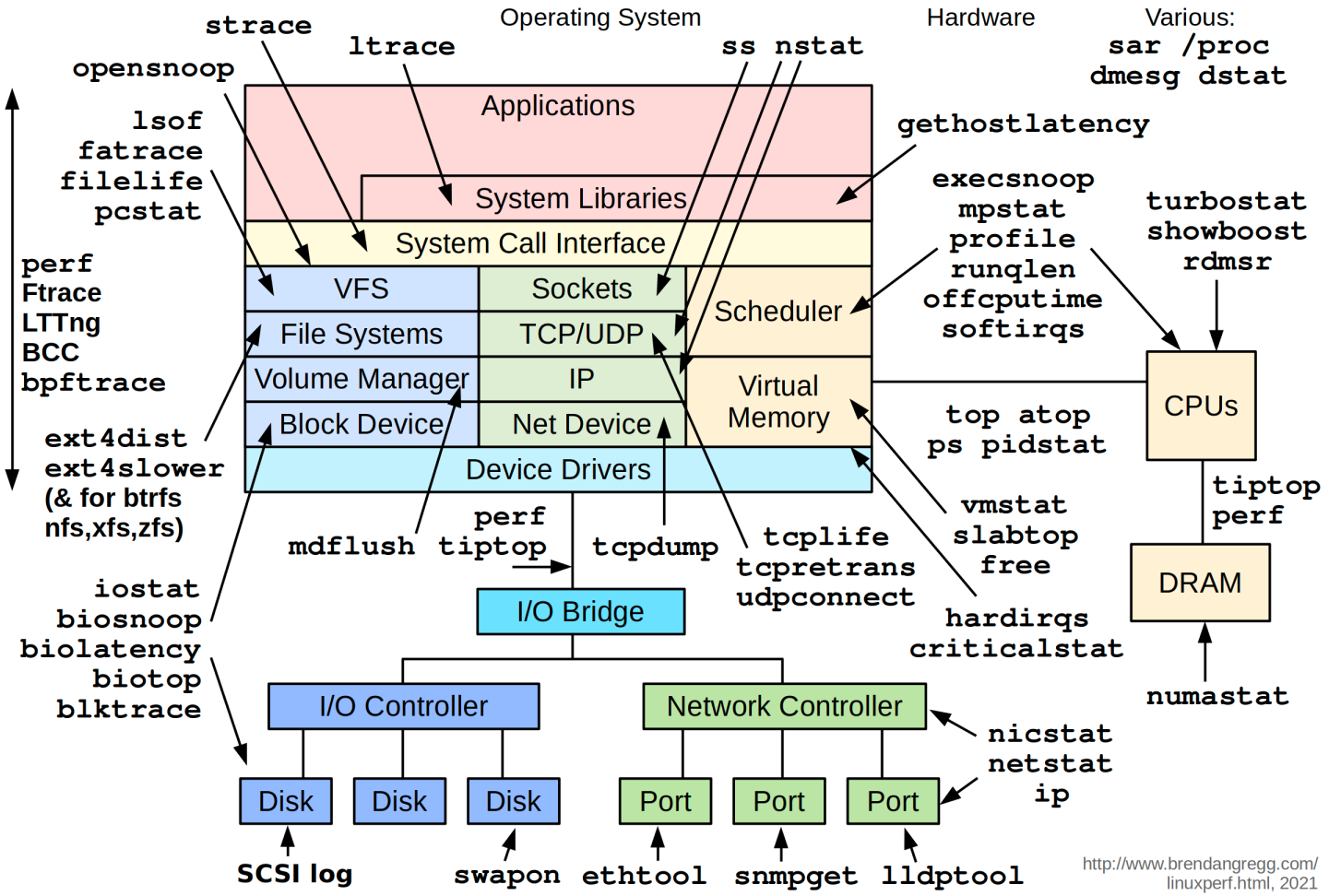
Próximos Passos:

- Explorar a API do kernel mais profundamente.
- Investigar técnicas avançadas de programação de módulos (ex: drivers).

Material Adicional

- The mind behind Linux | Linus Torvalds | TED YouTube
- Kernel Dev From Scratch YouTube
- Loadable Kernel Modules - basic introduction and tutorial of module commands YouTube
- Two decades of Git: A conversation with creator Linus Torvalds YouTube
- Steven Rostedt - Learning the Linux Kernel with tracing YouTube
- Linux Device Drivers, Third Edition

Linux Performance Observability Tools



Fonte da Imagem: Linux Perf

Dúvidas e Discussão