Sistemas Operativos – 2025/2026

Engenharia Informática - ESTiG / IPB

Trabalho Prático 1

1 Objetivos

No final do trabalho deverão ter sido atingidos os seguintes objetivos gerais: 1) saber como reconfigurar, recompilar e instalar um novo *kernel* do Linux; 2) saber como adicionar primitivas ao *kernel* e tirar partido das mesmas em pequenos programas.

2 Desenvolvimento

2.1 Pré-Requisitos

O trabalho deve ser realizado numa máquina virtual Linux baseada na distribuição Debian GNU/Linux (64 bits, versão 13.1 ou superior), com todas as atualizações aplicadas e a correr um *kernel* de versão 6.12.43 ou superior, de forma a garantir um ambiente uniforme para todos os alunos. A instalação deve ser não gráfica (modo texto), incluindo apenas as funcionalidades padrão do sistema e excluindo explicitamente qualquer ambiente gráfico de trabalho, de modo a manter uma configuração mínima e consistente adequada à compilação e experimentação do *kernel*.

A máquina virtual deve ser configurada com, pelo menos, 4 GB de RAM, 2 núcleos de CPU e 32 GB de espaço em disco. A máquina virtual fornecida no âmbito da unidade curricular pode ser utilizada, devendo, no entanto, o disco ser expandido. Sempre que possível, recomenda-se a atribuição de recursos de hardware adicionais, nomeadamente núcleos de processamento, uma vez que a recompilação do *kernel* Linux pode ser bastante exigente. Os alunos podem também optar por configurar a sua própria máquina virtual de raiz, desde que cumpram os requisitos do sistema operativo especificados (nomeadamente tratar-se de uma instalação exclusivamente em modo de linha de comandos).

Nota: Para expandir a capacidade de disco da máquina virtual fornecida no âmbito da disciplina de Sistemas Operativos, pode utilizar a ferramenta GParted (versão Live CD). Consulte tutoriais disponíveis online para orientação.

Importante: Um disco virtual de 32 GB é suficiente para o trabalho, desde que não se compile a totalidade dos módulos do *kernel*. De facto, neste trabalho, isso não é necessário: na fase 1, apenas será preciso desativar o módulo de suporte ao IPv6; na fase 2, as alterações a fazer ao *kernel* não implicam sequer lidar com módulos. Caso não se consiga descobrir a forma de otimizar a compilação do *kernel*, será melhor expandir o disco virtual para 64 GB, a fim de suportar a compilação total do *kernel*.

Nota: Se os computadores pessoais do grupo não tiverem capacidade suficiente para os requisitos do trabalho, deverá ser usado um computador fixo da escola (preferencialmente do Laboratório de Computação Avançada (LCA)), acedido com a conta pessoal (e não com a conta "aluno") de um dos elementos do grupo.

Nota: Durante a realização do trabalho poderão acontecer situações em que a máquina virtual deixa de arrancar ou fica num estado indesejado; assim, aconselha-se a criar um *snapshot* (uma espécie de cópia leve da máquina virtual), antes de realizar operações potencialmente problemáticas, para que seja possível regressar ao estado anterior da máquina virtual e evitar ter de a reinstalar na totalidade.

2.2 Fase 1

Na Fase 1 do trabalho pretende-se i) ficar a conhecer algumas das possibilidades de configuração do *kernel*, através da exploração do seu menu de configuração, ii) desativar o suporte para IPv6 através do referido menu, produzindo uma nova configuração, iii) compilar um novo *kernel* com base na nova configuração, e iv) instalar o novo *kernel* de forma a garantir a sua execução após reiniciar a máquina virtual.

Note-se que a compilação de um *kernel* completo pode ser muito demorada (levando horas) e consumir muito espaço em disco (vários Gigabytes). Por isso é conveniente adotar estratégias para reduzir o tempo de compilação e o espaço necessário, como: a) compilar o menor número possível de módulos (idealmente, só os módulos em uso); b) compilar em paralelo, tirando partido dos vários núcleos de CPU da máquina virtual; c) aumentar o número de núcleos de CPU e a RAM da máquina virtual. A adoção da estratégia a) e/ou b) será valorizada, desde que documentada no relatório.

Depois da máquina virtual arrancar com o novo *kernel*, devem-se executar os comandos `uname -r` e `cat /proc/net/if_inet6`; o 1.º permite verificar que o nome do *kernel* em execução está relacionado com o nome que se escolheu durante o processo de compilação; o 2.º permite comprovar que o *kernel* deixou de suportar IPv6 (ou seja, foi eficaz a desativação dessa funcionalidade na reconfiguração do *kernel*).

Nota: Para facilitar o acesso ao menu de arranque da máquina virtual, de forma a escolher a versão desejada do *kernel*, editar o ficheiro de configuração desse menu (`sudo nano /etc/default/grub`), definir o valor `GRUB_TIMEOUT=10` (menu disponível por 10s), gravar o ficheiro, aplicar esta alteração (`sudo update-grub`) e reiniciar a máquina virtual; durante o arranque pressionar a tecla SHIFT para aceder ao menu.

2.3 Fase 2

Na Fase 2 serão efetuadas as alterações necessárias ao *kernel* produzido na Fase 1 (iniciar a Fase 2 só após concluir a Fase 1), a fim de suportar duas novas primitivas:

- `long gcd(long a, long b, int debug)`: retorna o máximo divisor comum (gcd = *greatest common divisor*) dos números inteiros `a` e `b`;
- `long getnchildren(int debug)`: retorna o número de processos filhos que o processo invocador tem, no momento em que a primitiva é invocada;

O parâmetro `debug` controla (`0`/`1`) a produção de mensagens que são acrescentadas no ficheiro `/var/log/kern.log`, permitindo acompanhar a execução das primitivas (uma forma de o fazer é executar o comando `sudo tail -f /var/log/kernel.log` antes de iniciar o programa de teste, ou o comando `dmesg` durante essa execução). É obrigatório, para efeitos de avaliação, que cada mensagem comece com o número de aluno de um dos elementos do grupo.

Após reiniciar a máquina virtual com o novo *kernel*, deverá comprovar-se o funcionamento correto das novas primitivas. Para o efeito, deverá ser executado o programa:

```
// mysyscalls-test.c

#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <sys/types.h>
#include <mysyscalls.h>
#define NCHILDREN 10

int main(){
 long a, b, c;
 scanf("%ld %ld", &a, &b);
 printf("gcd(%ld,%ld,1)=%ld\n", a, b, gcd(a,b,1));

 int i, original_parent=getpid();
 printf("before fork: current number of children: %ld\n", getnchildren(1));
 for (i=0; i<NCHILDREN; i++) {
     if (fork()==0) {
         while (getppid() == original_parent);
         exit(0);
     }
 }
 printf("after fork: current number of children: %ld\n", getnchildren(1));

 return 0;
}
```

Neste programa, assume-se que `mysyscalls.h` terá o código necessário para a implementação da componente *userland* das duas primitivas, a qual basicamente invoca a componente no *kernel* (ou seja, assim como para poder invocar `fork` basta ter `#include <unistd.h>`, então para poder invocar as novas primitivas deverá ser suficiente fazer-se `#include <mysyscalls.h>`). Levar ainda em conta que se o ficheiro `mysyscalls.h` estiver localizado, genericamente, na pasta `z` de caminho absoluto `/x/y/z` então, na compilação do programa acima, com o compilador `gcc`, deve-se usar a diretiva `-I/x/y/z` para que o compilador encontre o ficheiro `mysyscalls.h` .

3 Avaliação

A avaliação será baseada i) na demonstração da eficácia das alterações realizadas na Fase 1, ii) na demonstração da execução do programa de exemplo indicado na Fase 2, e iii) na defesa e análise de um pequeno relatório que documente os passos seguidos na Fase 1 e na Fase 2 (neste caso, o relatório deve conter obrigatoriamente o código fonte do ficheiro `mysyscalls.h` e as várias modificações feitas a ficheiros do *kernel*, incluindo também novos ficheiro que se tenham criado). O relatório deve basear-se, necessariamente, no modelo publicado juntamente com este enunciado (o modelo tem já a estrutura esperada e indicações sobre o conteúdo a adicionar). Se a máquina apresentada na defesa não refletir as ações descritas no relatório, não será atribuída qualquer pontuação ao grupo.

4 Grupos e Prazos

O trabalho deverá realizar-se em grupo, constituído por 2 elementos (apenas se permitindo exceções para estudantes trabalhadores ou outros casos autorizados).

O relatório (em formato PDF) deverá submeter-se através do sítio virtual.ipb.pt, na área de Atividades de Sistemas Operativos, até às 24h de 30/11/2025. Se esta funcionalidade estiver indisponível, ou se a submissão não for bem sucedida, o relatório poderá enviar-se por e-mail, para `arnaldo@ipb.pt`, até às 0h15m do dia 1/12/2025.

A defesa (demonstração e avaliação oral) do trabalho será realizada em data a divulgar oportunamente. As classificações dos elementos de cada grupo poderão ser diferenciadas, mediante o desempenho individual durante a defesa do trabalho.

5 Sugestões

Para compilar um novo *kernel* de forma a suportar não mais que os módulos em uso pelo *kernel* atual (ou até menos, caso se desativem alguns, como será o caso do módulo do IPv6), existe um comando do tipo `make ...`, que é necessário descobrir.

O termo *system call* (ou, mais abreviadamente, *syscall*) é a designação habitualmente usada, em inglês, para o conceito de *chamada ao sistema* ou *primitiva*.

Em relação à primitiva `getnchildren`, o capítulo 3 (Process Management) do livro "Linux Kernel Development", Robert Love, Addison-Wesley, 2010 poderá ser útil.