

Linux

Obsah

1	Úvod	1
2	Konfigurace jádra	2
3	Překlad jádra	3
4	Adresářová struktura	4

1 Úvod

2 Konfigurace jádra

Než je možné jádro Linuxu přeložit je třeba provést konfiguraci, která umožní přizpůsobit chování systému pro daný hardware, zvýšit výkon zmenšit velikost výsledného binárního souboru (zakázáním nepotřebných modulů).

Konfiguraci jádra lze rozdělit do několika bodů:

1. **Architektura a vlastnosti procesoru (Processor type and features)** - absolutní základ, který je třeba nastavit pro správné fungování na dané platformě:
 - **Typ procesoru** - například intel Core, AMD ryzen, ARM, ...
 - **Podpora vícejádrových procesorů** - pokud má daný procesor více jader, umožní plánovači systému efektivně rozdělovat jádra běžícím procesům
 - **Bezpečnostní funkce procesoru** - tyto volby zvyšují bezpečnost výsledného jádra, jedná se o volby jako randomizace RAM adres nebo ochrana zásobníku proti přetečení
2. **Obecné nastavení jádra** -

2.1 Obecné nastavení (General configuration)

2.1.1 Komprimace jádra (Kernel compression mode)

Kernel compression mode je nastavení v konfiguraci Linuxového jádra, které určuje, jakým kompresním algoritmem bude výsledný soubor jádra (tzv. kernel image, obvykle vmlinuz nebo bzImage) po kompilaci komprimován. To že je jádro na disku v komprimované podobě umožňuje úsporu paměti disku, rychlejší bootování, menší využití RAM nebo rychlejší síťové bootování. Když vyberete kompresní algoritmus a zkompilujete jádro, kód dekompresoru pro vybraný algoritmus je zabudován přímo do počáteční části jádra. Když bootloader (např. GRUB) načte komprimované jádro do paměti a předá mu řízení, první věcí, kterou jádro udělá, je dekomprese sebe sama do paměti. Teprve potom začne spouštět zbytek systému.

Jádro Linuxu podporuje několik kompresních algoritmů, které se liší kompresním poměrem a rychlostí dekomprese. Základní volby jsou

- **GZip** - dobrý kompresní poměr, ale pomalejší dekomprese
- **BZip2** - lepší kompresní poměr než GZip ale výrazně pomalejší
- **LZMA** - velice dobrý kompresní poměr ale zároveň velice pomalá dekomprese, vhodný pro embedded systémy s omezenou pamětí
- **XZ** - podobné vlastnosti jako LZMA
- **LZO** - nižší kompresní poměr, ale velice rychlá dekomprese, vhodné pro systémy s potřebou rychlého bootování, kde velikost jádra je méně důležitá
- **LZ4** - ještě nižší kompresní poměr ale ještě rychlejší dekomprese než LZO, vhodné pro desktopové systémy

2.1.2 Preemption Model

Model preemptibility jádra (anglicky Kernel Preemption Model nebo též Preemption Model) je klíčové nastavení Linuxového jádra, které ovlivňuje jeho odezvu (responsivitu) a propustnost (throughput). Určuje, kdy může jádro přerušit (preemptovat) běžící kód jiného procesu nebo samotného jádra, aby se mohla spustit úloha s vyšší prioritou.

V kontextu operačních systémů znamená preempece možnost operačního systému odejmout procesoru kontrolu nad běžící úlohou (procesem nebo kódem jádra) a přidělit ji jiné úloze, obvykle té s vyšší prioritou, nebo jednoduše té, která čeká na vykonání.

V jádře Linuxu se preempece týká kódu běžícího v jádrovém režimu (kernel mode). Tradičně, když se kód jádra spustí, běží, dokud sám neskončí, nebo dokud nenarazí na

blokující operaci (např. čekání na I/O). Během této doby nemůže být přerušen jinou úlohou, dokonce ani tou s vyšší prioritou. To zajišťuje jednoduchost a minimalizuje riziko složitých stavů, ale na úkor odezvy.

S preempčním jádrem se situace mění. Jádro může být nakonfigurováno tak, aby se chovalo jako "klient" pro scheduler, což umožňuje přerušování (preemptování) kódu jádra jiným, naléhavějším úkolem.

Volba modelu preemptibility je kompromis mezi:

- **Odezvou (Responsiveness):** Jak rychle systém reaguje na externí události (např. stisk klávesy, síťový paket, multimediální data). Dobrá odezva je klíčová pro desktopové systémy, audio/video aplikace a real-time systémy.
- **Propustností (Throughput):** Celkové množství práce, které systém zvládne za daný čas. Vyšší propustnost je často prioritou pro servery, databázové systémy nebo HPC (High Performance Computing), kde je cílem dokončit co nejvíce úloh, i za cenu mírně delších latencí.

Obecně platí **Více preempce = lepší odezva**, ale potenciálně mírně nižší propustnost kvůli vyšší režii (overhead) z neustálého přepínání kontextu. **Méně preempce = vyšší propustnost**, ale potenciálně horší odezva.

Dostupné nastavení v jádru:

- **No Forced Preemption (Server)** - nejméně preemptivní model, kód jádra běží nepřerušovaně, dokud sám nedokončí, nebo dokud nenarazí na definovaný bod přerušování (čekání na IO, explicitní volání funkce `shedule()`). Ideální pro servery, databáze, HPC, kde je prioritou na maximální propustnost a stabilní, předvídatelný běh s minimálním rušením.
- **Voluntary Preemption (Desktop)** - kompromisní model. Kód jádra má do sebe vložený dodatečné "body přerušování". Jádro si kontroluje, zda je proces s vyšší prioritou připraven ke spuštění, a pokud ano, dobrovolně se vzdá procesoru. Dříve běžná volba pro desktopové systémy, než se rozšířila plná preempce.
- **Preemptible Kernel (Low-Latency Desktop)** - toto je nejvíce preemptivní model pro běžné použití. Většina kódu jádra je plně preemptibilní, což znamená, že může být přerušena kdykoli jiným úkolem s vyšší prioritou (nebo jen jiným úkolem, který potřebuje CPU). Jádro se chová mnohem více jako normální uživatelské procesy. Standardní a doporučená volba pro desktopové systémy, multimediální stanice a interaktivní pracovní stanice.
- **Scheduler controlled preemption model** - Místo, aby se rozhodovalo o preemptibilitě celého jádra při kompilaci staticky, tato volba přesouvá část rozhodování o preempci na běhový plánovač (scheduler). Jádro může dynamicky rozhodovat o tom, jak agresivně bude preemptovat, a to na základě zatížení systému, typu běžících úloh a dokonce i na základě konfiguračních parametrů, které lze nastavit za běhu.

3 Překlad jádra

4 Adresářová struktura