# 12. Unix

**01** Charakteristika systému

- **a)** Srovnání s MS Windows
- **b)** Adresářová struktura

**02** Uživatelský a programátorský interface

- **a)** Základní příkazy, roury, tvorba skriptů, systémové proměnné, vrstvy

**03** Použití a popis služeb

- **a)** Telnet, SSH, FTP, DNS, DHCP

**04** Vizualizace Unixového prostředí na MS Windows

# 12. Unix

## Charakteristika systému

**Základní informace**

- UNIX je **víceuživatelský a víceúlohový univerzální operační systém**.
- Byl vyvinut ve výzkumných laboratořích **AT&T** Bell Labs – autoři: Ken Thompson a Dennis Ritchie *(přelom 60.–70. let)*.
- Většina systému UNIX je napsána v **jazyce C**, což **zajišťuje přenositelnost mezi různými hardware platformami**.

**Souborový systém**

- **UNIX má hierarchický souborový systém:** stromová struktura adresářů s jedním kořenem *(/)*.
- **Každé zařízení je reprezentováno jako soubor** – jednotný přístup ke všem prostředkům *(např. /dev/sda, /dev/tty)*.

**Víceuživatelský a víceúlohový systém**

- **Víceuživatelský režim:** umožňuje současný přístup více uživatelům *(např. přes terminály nebo vzdálené přihlášení)*.
- **Víceúlohový režim** *(multitasking)*: každý uživatel může spouštět více procesů současně.

**Úlohy operačního systému**

- **Správa paměti, procesů, disku a vstupně-výstupních** zařízení.
- Poskytuje prostředí pro **běžnou práci uživatele i vývoj aplikací**.

**Platformová nezávislost**

- UNIX běží na široké škále zařízení – **od superpočítačů až po PC.**
- Je **velmi rozšířený na výkonných** pracovních stanicích.

**Softwarová vybavenost**

- Dodává se s množstvím programového vybavení:
  - Textové editory *(např. vi, WordPerfect)*
  - Kompilátory *(C, C++, Fortran…)*
  - Databázové systémy *(např. dBASE, Oracle)*
  - CAD nástroje *(např. AutoCAD, OrCAD)*
  - Open-source programy *(např. X Window, TeX, GCC)*

**Vlastnosti systému UNIX**

- **Jednoduché uživatelské rozhraní** – příkazová řádka *(shell)* a **volitelně i grafické prostředí** *(X11)*.
- **Zařízení jsou prezentována jako soubory** – jednotný přístup *(např. čtení/zápis do tiskárny jako do souboru)*.
- **Silný shell** *(příkazový jazyk)* – shell je **plnohodnotný skriptovací jazyk** s podporou:
  - Cykly, podmínky, proměnné, funkce
  - Spojování programů pomocí **rour** *(pipes)* a přesměrování
- Nástroje **pro práci s textem a soubory**:
  - Vyhledávání, porovnávání, třídění, filtrování, práce s tabulkami, rovnicemi
- **Vývojové nástroje:**
  - Kompilátory, debuggery, profilery
  - Lexikální a syntaktické analyzátory *(např. lex, yacc)*
  - Podpora týmové spolupráce

# 12. Unix

**Síťové funkce**

- UNIX obsahuje **základní síťové programové vybavení**:
  - Podpora protokolů **TCP/IP**
  - Připojení do počítačových sítí
  - Síťové nástroje a služby *(viz pozdější sekce: Telnet, SSH…)*

**Shrnutí pro zapamatování:**

- UNIX = **přenosný, víceuživatelský, víceúlohový OS**
- Napsán v jazyce C → **běží skoro všude**
- **Silný shell**, hierarchický systém, jednotné rozhraní pro vstup/výstup
- **Bohatá výbava nástrojů** pro práci i vývoj
- **Silná síťová integrace**

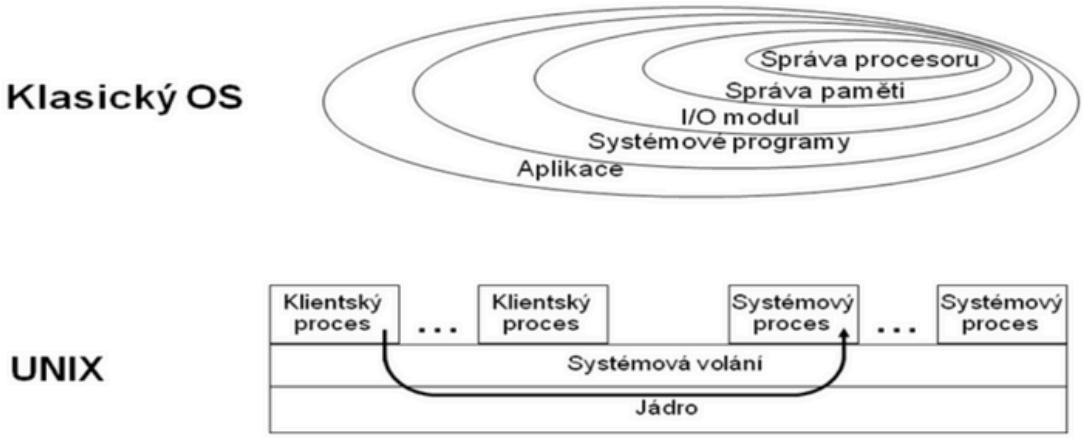
## UNIX – Srovnání s MS Windows

| Vlastnost | UNIX/Linux | Windows |
|---|---|---|
| **Licencování** | Open Source – uživatel má přístup ke zdrojovému kódu, může ho upravovat a přizpůsobit systému. | Proprietární (uzavřený kód) – zdrojový kód je nepřístupný, chráněný licencí Microsoftu. |
| **Zabezpečení** | Vysoká bezpečnost – otevřený kód umožňuje rychlé odhalení a opravu chyb, nižší výskyt virů. | Častý cíl útoků – velká uživatelská základna, více malwaru a bezpečnostních děr. |
| **Výkon a nároky** | Nižší systémové nároky, běží rychle i na starším hardware. | Vyšší nároky na výkon, pomalejší na slabších strojích. |
| **Způsob přístupu k zařízení** | Všechna zařízení (např. HDD, tiskárny, CD-ROM) jsou reprezentována jako soubory (např. /dev/sda, /dev/lp0). | Zařízení jsou spravována jako objekty nebo jednotky (např. C:, D:, E:, USB Drive). |
| **Adresářová struktura** | Stromová struktura začínající kořenovým adresářem / (např. /home, /etc, /bin). | Soubory jsou rozděleny podle písmen jednotek (C:\, D:\, atd.), adresáře = složky. |
| **Duplicitní názvy** | Nelze mít dva soubory se stejným názvem ve stejném adresáři – to platí pro obě platformy. (Pozor: uvedené tvrzení o Linuxu je nesprávné – systém nedovolí duplicitu ve stejné složce, stejně jako Windows.) | Stejné – nelze mít duplicitní název souboru ve stejné složce. |
| **Uložení systémových a programových souborů** | Různě oddělené – systémové soubory ve /etc, /bin, /sbin, programové v /usr, /opt, atd. | Většina systémových a programových souborů je umístěna na jednotce C:\ (např. C:\Windows, C:\Program Files). |

# 12. Unix

**Shrnutí pro zapamatování**

- UNIX/Linux je **otevřený, rychlý, bezpečný a univerzální systém** s jednotným přístupem k zařízení jako k souborům.
- Windows je **uzavřený, běžnější systém s jednodušším GUI**, ale **vyššími nároky a větší náchylností k útokům.**
- Hlavní rozdíly spočívají v **přístupu k zařízení, způsobu správy systému, otevřenosti a systémové struktuře.**




## Práce se soubory

**Co je soubor v UNIXu?**

- Soubor = **sekvence** *(sled)* **bajtů**. UNIX nerozlišuje mezi různými typy obsahu – všechny soubory jsou pro jádro **„jen data“.**
- Soubory se rozlišují **na základě použití** – např. **textové, binární, zařízení, adresáře atd**.

**Textové soubory**

- Textové programy *(např. cat, less, editory)* očekávají, že **řádky jsou ukončeny znakem ASCII LF** *(Line Feed, kód \n, hex 0A)*.
- **Pozor:** UNIX používá **jen LF**, zatímco Windows používá CR+LF *(\r\n, hex 0D0A)*.

**Adresářová struktura** *(stromová hierarchie)*

- UNIX používá **hierarchický souborový systém se strukturou stromu**, který začíná **kořenovým adresářem /**.
- Adresáře jsou také soubory, které **obsahují informace o tom, kde se nachází jiné soubory** – např. název a inode *(interní identifikátor)*.

### Cesty k souborům

K souborům můžeme **přistupovat pomocí cest** *(path):*

1. **Relativní cesta**

- Vztahuje se k aktuálnímu adresáři *(working directory)*.
- **.** *(tečka)* označuje **aktuální adresář**
  - Např. **./trida/lavice** → cesta z aktuální složky do **podadresáře trida**, dále lavice.
- **..** *(dvě tečky)* označuje **nadřazený adresář**
  - Např. **../škola/soubor.txt** → o složku výš, pak do škola, a pak soubor.

**Příklad:**

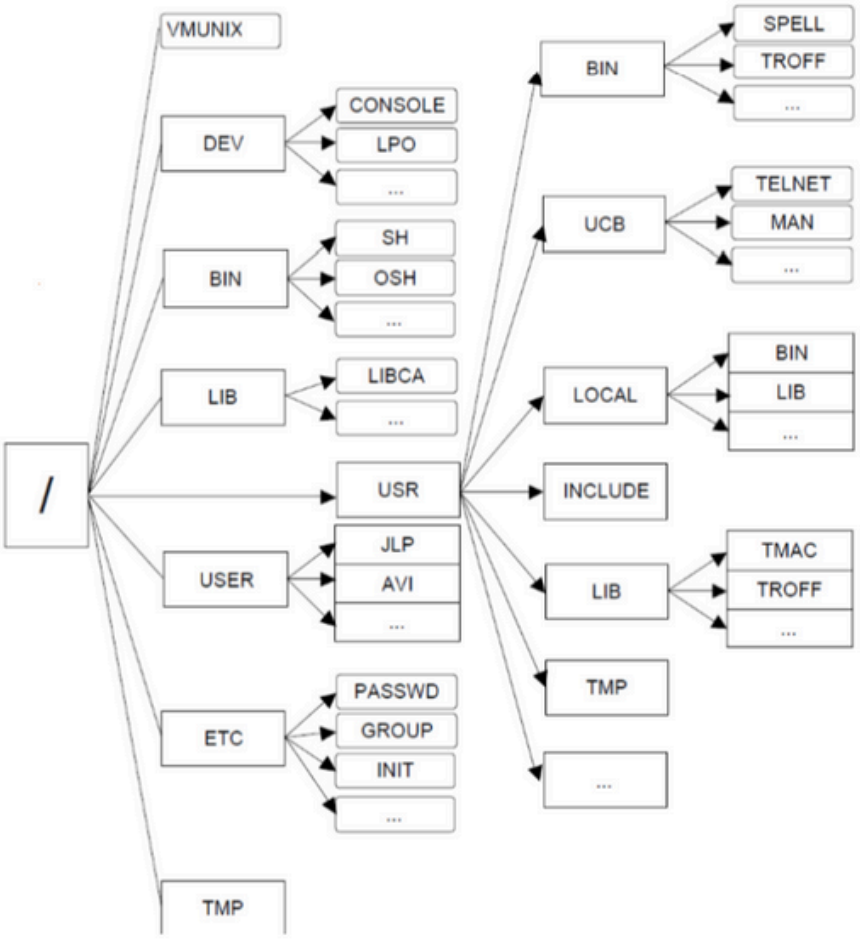
- Pokud jsem v **/home/user/dokumenty/**
  - ./trida/lavice ➔ /home/user/dokumenty/trida/lavice
  - ../škola/soubor.txt ➔ /home/user/škola/soubor.txt

2. **Absolutní cesta**

- Vztahuje se vždy **od kořene systému /**, nezávisle na aktuálním adresáři.
- Např. **/home/x0a_gorny/škola/soubor.txt**
- **Výhoda:** přesná a jednoznačná
- **Nevýhoda:** často dlouhá, méně přehledná

**Shrnutí pro zapamatování**

- Soubor = sled bajtů, bez vnitřní struktury
- Řádky v textu odděleny znakem **LF (\n)**
- **Adresáře jsou také soubory** – tvoří stromovou strukturu
- **Cesty:**
  - **Relativní:** závisí na **aktuální pozici** *(. a ..)*
  - **Absolutní:** od kořene **/**, univerzální ale dlouhá



# 12. Unix

## Programátorský interface (Systém souborů v UNIXu)

### Struktura a organizace

- UNIX používá **stromovou hierarchii adresářů**, kde jsou všechny soubory i zařízení umístěny v jednom souborovém systému.
- Kořenový adresář je **/** – od něj vede celá struktura.
- Adresáře mohou **obsahovat soubory i další podadresáře.**
- **Zařízení** *(např. tiskárny, pevné disky)* jsou reprezentována jako **speciální soubory ve stromu** *(typicky v /dev)*.

### Absolutní vs. relativní cesta

- **Absolutní cesta:**
  - Začíná **/** *(kořenem)*
  - Určuje **jednoznačnou polohu souboru v systému**
  - Např. **/home/uzivatel/dokument.txt**
- **Relativní cesta:**
  - Nezačíná lomítkem
  - Vztahuje k tzv. **pracovnímu adresáři** *(working directory)* procesu
  - Např. **dokument.txt** nebo **../soubory/data.txt**

### Další vlastnosti:

- UNIX rozlišuje **malá a velká písmena** *(file.txt ≠ File.txt)*.
- Jméno souboru může **obsahovat libovolné znaky, včetně mezer** *(i když to není doporučeno)*.
- **Maximální délka názvu souboru:**
  - Historicky 14 znaků
  - Dnes typicky až 255 znaků
- Každý běžící proces má **přiřazený aktuální adresář**, který se dá zjistit např. **příkazem pwd.**

| Adresář | Popis |
|---|---|
| **/dev** | Obsahuje soubory reprezentující zařízení systému (disky, tiskárny, terminály). |
| **/bin** | Základní spustitelné příkazy systému (např. ls, cp, mv, cat). |
| **/sbin** | Systémové příkazy určené pro administrátory (root) (např. fsck, reboot). |
| **/etc** | Konfigurační soubory a skripty, např. nastavení sítě, start serveru. |
| **/home** | Domovské adresáře běžných uživatelů (např. /home/riso). |
| **/lib** | Sdílené knihovny, především pro jazyk C a další systémové programy. |
| **/tmp** | Dočasné soubory, které se po restartu obvykle mažou. |
| **/var** | Proměnlivé soubory – logy, fronty tiskáren, e-maily apod. |
| **/mnt** | Připojovací bod pro externí disky nebo oddíly (mount point). |
| **mc -a** | Spuštění Midnight Commanderu – dvoupanelový správce souborů v terminálu. |

# 12. Unix

## Procesy

- Proces = **spuštěný program** *(instance programu běžící v paměti)*.
- Každý proces má **unikátní identifikátor** *(PID – Process ID)*.
- Proces má vlastní:
  - **Adresní prostor** *(paměť)*
  - Otevřené soubory
  - **Stav** *(běžící, čekající, zombie…)*
  - **Prioritu a rodiče** *(PPID – Parent PID)*

## Vztah rodič-potomek *(forkovací model)*

**fork()**

- Systémová funkce, která **vytváří nový proces** *(kopii původního)*.
- **Potomek dědí celý adresní prostor rodičovského procesu**.
- Oba procesy *(rodič i potomek)* pokračují od **stejného místa v kódu za voláním fork()**.
- Funkce **fork() vrací:**
  - **0** v potomkovi
  - **PID** potomka v rodiči
  - **–1 při chybě** *(např. Cannot fork: Resource temporarily unavailable)*

**exec()**

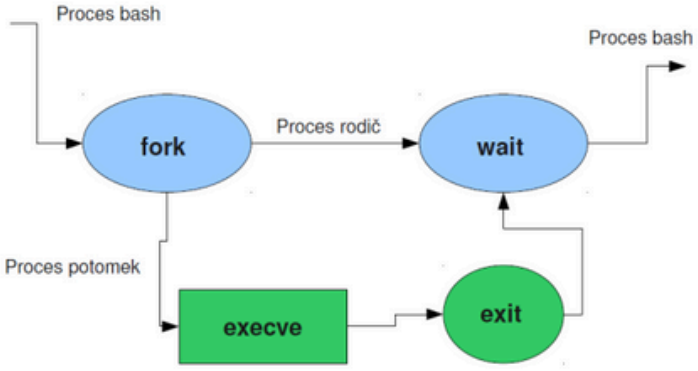
- Zavádí **nový program do adresního prostoru procesu** *(většinou potomka)*.
- **Přepisuje aktuální kód a data procesu** – původní program končí.
- Často používané varianty: **execl(), execv(), execvp()…**

**exit()**

- Potomek pomocí této funkce **ukončuje svou činnost**.
- Vrací návratovou hodnotu, kterou může zachytit rodič *(např. úspěch/neúspěch)*.
- Proces poté přechází do stavu **ZOMBIE** – zůstává v systému, **dokud ho rodič nesklidí** *(wait())*.

**wait()**

- **Rodič čeká na ukončení potomka.**
- **Funkce vrátí PID** ukončeného potomka a **umožňuje zpracovat jeho návratový kód**.
- Pokud rodič nezavolá **wait()**, zůstane potomek ve stavu zombie.



Vznik nového procesu a start nového programu

**Životní cyklus procesu**

1. **Vytvoření** *(fork)*
2. *(volitelně)* **Nahrazení** kódu *(exec)*
3. **Běh** procesu
4. **Ukončení** *(exit)*
5. **ZOMBIE stav** – dokud rodič nevolá wait


**Shrnutí pro zapamatování**

- **Proces** = běžící program, má PID, vlastní paměť a stav.
- **fork():** vytvoří nový proces *(kopie rodiče)*
- **exec():** načte nový program *(přepíše proces)*
- **exit():** ukončí proces a předá návratovou hodnotu rodiči
- **wait():** rodič čeká na ukončení potomka
- Nevyzvednutý potomek zůstává jako **zombie**

# 12. Unix

## Vznik procesu & PIPE (roura)

### Plánování *(scheduling)* a priorita

- Každý **proces má prioritu**, která ovlivňuje, **kdy a jak často bude procesorem spuštěn**.
- Priorita je číselná hodnota, kladné číslo, které je jedním z faktorů plánování.
  - **Nižší číslo = vyšší priorita** *(čím "nicer", tím méně náročný = nižší priorita)*.

**Dědění a nastavení priority**

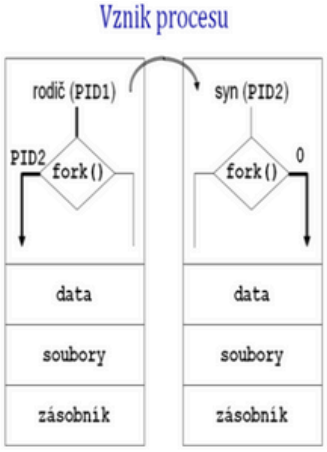
- Potomek **dědí prioritu rodiče** při vytvoření pomocí **fork().**
- **Lze ale nastavit jinou prioritu** při startu pomocí příkazu:
  - **bash:** nice -n [incr] [command]
- **incr** je **inkrement priority**, typicky v rozsahu -20 *(nejvyšší priorita)* až +19 *(nejnižší)*.
- Pouze **root může zadat záporné** hodnoty *(vyšší prioritu)*.
  - **příklad:** nice -n 10 ./script.sh  # spustí script s nižší prioritou

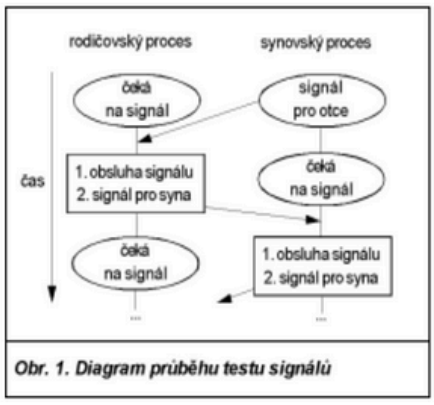
**Změna priority běžícího procesu**

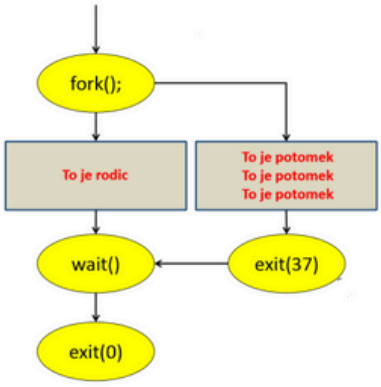
- **Pomocí příkazu:** renice -n [incr] [PID]
- Slouží **ke změně priority již běžícího** procesu dle jeho PID.
  - **Příklad:** renice -n -5 1234  # zvýší prioritu procesu s PID 1234 *(povoleno jen rootem)*

**Zabití procesu a stav zombie**

- **Po ukončení (exit())** se proces označí jako **ZOMBIE** *(Z stav v ps)*.
- **Zombie proces:**
  - Je **ukončený**, ale stále evidovaný v systému *(čeká na wait() rodiče)*.
  - **Nemůže být znovu spuštěn.**
  - Zůstává v paměti jen **kvůli návratovému statusu.**




## PIPE – Roura (meziprocesová komunikace)

**Co je PIPE?**

- **Systémová vyrovnávací paměť** *(buffer)*, pomocí které **jeden proces zapisuje výstup a druhý ho čte jako vstup.**
- **Používá se k propojení příkazů** – standardní výstup *(stdout)* jednoho slouží jako standardní vstup *(stdin)* druhého.

**Syntaxe:**

- command1 | command2
- Příklad: **ls -l | more** *(vypíše seznam souborů, more ho zobrazí stránkovaně)*
  - command1 ➔ **generuje data** *(výstup)*
  - command2 ➔ **zpracovává data** *(vstup)*

# 12. Unix

**Kolona**

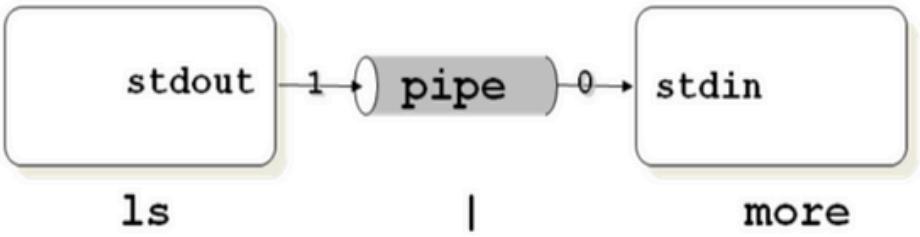
- **Sekvence více příkazů spojených rourami** se nazývá kolona *(pipeline)*.
- **Může obsahovat více členů**, přičemž každý předává výstup dalšímu.
- **Příklad:** ps aux | grep apache | sort -k2

**Shrnutí pro zapamatování:**
**Vznik procesu:**

- **fork()** – vytvoření potomka, dědí prioritu
- **nice** – nastaví prioritu při spuštění
- **renice** – změní prioritu běžícího procesu
- **Nižší nice** = vyšší priorita, pouze root může dát záporné číslo
- **ZOMBIE** = ukončený, neodklizený proces

**PIPE:**

- **Propojení procesů** pomocí **|**
- **Data tečou z výstupu** jednoho **do vstupu druhého**
- Víc příkazů = **kolona** *(pipeline)*




## UNIX – Meziprocesová komunikace (IPC – Inter Process Communication)

Meziprocesová komunikace **umožňuje výmenu dat a synchronizaci mezi procesy**, které běží současně. **Existuje více metod:**

### 1. Semafory

Semafor slouží k **synchronizaci přístupu k tzv. kritické sekci** – části programu, která **pracuje se sdíleným zdrojem** *(např. tiskárna, paměť, soubor)*.
**Typy přístupu:**
**a) Zprostředkovatel** *(proxy)*

- Ke zdroji **nepřistupuje přímo žádný program**.
- Přístup je **řízen přes samostatný proces**, který spravuje přístup ostatním.
- **Výhoda:** vyšší kontrola, přehlednost.

**b) Detektivní algoritmus** *(busy-waiting)*

- Procesy **opakovaně kontrolují přepínač** *(semafor)*, zda je zdroj volný.
- Při víceúlohovém prostředí je to **neefektivní → zbytečné zatížení CPU.**
- Nebezpečí tzv. **kruhového blokování** *(deadlock)* – **procesy čekají navzájem na uvolnění zdroje.**

**c) Dijkstrův semafor** *(klasický semafor)*

- Speciální **celočíselná proměnná** (kladné hodnoty)
- Používá **dvě základní atomické operace:**
  - **P(A)** *(prolož/čekej)* **-** Pokud **A > 0**, sníží hodnotu A; pokud **A == 0**, proces čeká.
  - **V(A)** *(uvolni/signál)* **-** Pokud na **A čeká proces**, ten pokračuje; **jinak A++.**
- Typická **implementace pomocí jádra** nebo knihovny.
- **Možný problém: Deadlock** – procesy vzájemně čekají na uvolnění zdroje.

# 12. Unix

## 2. Sdílená paměť (Shared Memory)

- Umožňuje více **procesům přistupovat ke stejnému prostoru v paměti**.
- **Velmi efektivní způsob výměny dat** – rychlejší než jiné IPC metody *(např. sockety)*.
- Jeden proces **paměť alokuje a inicializuje**, ostatní ji "**namapují**" do svého prostoru.
- **Není automaticky synchronizovaná →** musí se kombinovat se semafory nebo zámky.
- Typicky se používá **shmget(), shmat(), shmdt()...**

**Problém:**

- Bez synchronizace může druhý proces číst nedokončený zápis, což může **vést k nekonzistenci dat.**

## 3. Signály

Signály jsou **asynchronní oznámení mezi procesy nebo z jádra procesům** – upozorňují je na určité události *(např. přerušení, chybu, ukončení...)*.

**Základní signály:**

| Signál | Význam | Obvykle vyvolán |
|---|---|---|
| **SIGINT** | Přerušení (Ctrl+C) | Uživatelem |
| **SIGQUIT** | Ukončení s výpisem jádra (Ctrl+\ nebo Ctrl+I) | Uživatelem |
| **SIGHUP** | Ukončení relace (např. zavření terminálu) | Shell |
| **SIGKILL** | Násilné ukončení procesu | Nelze zachytit nebo ignorovat |
| **SIGTERM** | Žádost o ukončení (standardní) | Lze zachytit |

**Použití:**

- **Poslání signálu:** kill -SIGNAL PID
- **Nejsilnější signál:** kill -9 PID  # SIGKILL
- **Zjištění PID procesu:** ps -a -l

## Shrnutí pro zapamatování

| Metoda | Popis | Použití |
|---|---|---|
| **Semafory** | Synchronizace přístupu ke zdrojům | P() a V(), pozor na deadlock |
| **Sdílená paměť** | Rychlá výměna dat | Vyžaduje synchronizaci |
| **Signály** | Oznámení o události | kill, SIGINT, SIGKILL atd. |

# 12. Unix

## UNIX – Uživatelský interface (rozhraní)

**Rozhraní systému**

- Uživatelský interface je **způsob, jakým uživatel komunikuje s operačním systémem.**
- **V UNIXu může být:**
  - **Textový** *(CLI)* – např. **shell** *(Bash, zsh)*
  - **Grafický** *(GUI)* – např. **GNOME, KDE** *(méně používaný na serverech)*

**Systémové programy** *(nástroje)*

- Programátoři i běžní uživatelé **využívají systémové programy**, často zaměřené na **manipulaci se soubory a adresáři**.

| Příkaz | Popis |
|---|---|
| **mkdir** | Vytvoření nového adresáře |
| **rmdir** | Odstranění prázdného adresáře |
| **cd** | Změna aktuálního adresáře |
| **pwd** | Zobrazení aktuálního adresáře (absolutní cesta) |
| **ls** | Výpis obsahu adresáře |
| **cp** | Kopírování souboru/adresáře |
| **mv** | Přesunutí nebo přejmenování souboru/adresáře |
| **rm** | Odstranění souboru/adresáře (i rekurzivně: rm -r) |

## Shell (terminálové prostředí)

Shell je **interpreter příkazů – tedy proces, který:**

- **zprostředkovává spouštění** jiných programů
- **umožňuje skriptování**, **přesměrování vstupů/výstupů, použití proměnných**

**Fungování shellu:**

1. Uživatel zadá příkaz *(např. ls)*
2. **Shell prohledá tzv. $PATH** – seznam adresářů, kde **hledá spustitelné soubory**
   - **Obvykle obsahuje:** /bin, /usr/bin, /usr/local/bin
3. Pokud je soubor nalezen, **shell jej spustí jako nový proces** a po dobu jeho běhu **„čeká“**
4. Po ukončení příkazu **shell pokračuje**

**Spuštění programu** = zadání jeho **názvu** *(např. program)* nebo **cesty** *(./program, /usr/bin/program)*

# 12. Unix

## Často používané příkazy

| Příkaz | Funkce |
|---|---|
| **find <cesta> -name <soubor>** | Vyhledává soubor v adresářové struktuře |
| **ls** | Výpis obsahu adresáře |
| **pwd** | Aktuální pracovní adresář |
| **cd <adresář>** | Přesun do adresáře |
| **mkdir <název>** | Vytvoří adresář |
| **rmdir <název>** | Smaže prázdný adresář |
| **chmod <modifikace> <soubor>** | Změní práva k souboru (např. chmod 755 script.sh) |
| **grep <text> <soubor>** | Vyhledá řádky obsahující zadaný text |
| **cat <soubor>** | Zobrazí obsah souboru, případně spojí více souborů |
| **head -n <číslo> <soubor>** | Zobrazí prvních n řádků |
| **tail -n <číslo> <soubor>** | Zobrazí posledních n řádků |
| **diff <soubor1> <soubor2>** | Porovná soubory po řádcích |
| **cmp <soubor1> <soubor2>** | Porovná bajt po bajtu, vypíše první rozdíl |
| **sort <soubor>** | Setřídí obsah souboru podle abecedy nebo číselně |

## Shrnutí:

- **Uživatelské rozhraní UNIXu je primárně textové** – shell je klíčový pro správu systému.
- Systémové příkazy poskytují **silné nástroje pro správu souborů a adresářů**.
- Shell spouští programy **z cest definovaných v proměnné prostředí $PATH**.
- Pro efektivní práci v UNIXu je **nutné znát a používat tyto příkazy a principy**.

# 12. Unix

## Použití a popis síťových služeb

**Telnet** *(Telecommunication Network)*

- **Textový komunikační protokol** pro **připojení k vzdálenému počítači přes síť**.
- Pracuje na **aplikační vrstvě modelu TCP/IP**.
- **Typ spojení:** klient–server, využívá TCP transportní protokol, přenos je duplexní *(obousměrný)*.
- Přenáší **8bitové znaky** *(ASCII)*, bez šifrování – **nezabezpečený**.
- Server naslouchá na portu 23.
- Umožňuje **ovládání vzdáleného počítače přes příkazový řádek**.
- Probíhá tzv. **vyjednávání voleb** – nastavení typu terminálu, ovládání toku atd.
- **Nevhodný pro používání v otevřených sítích** *(např. Internet)* kvůli bezpečnostním rizikům – hesla a data nejsou šifrována.

**SSH** *(Secure Shell)*

- **Zabezpečený nástupce Telnetu** pro vzdálený přístup a správu systému.
- Pracuje na **aplikační vrstvě TCP/IP**, používá TCP port 22.
- **Šifruje celý přenos:** chrání hesla, příkazy, výstup i data.
- **Umožňuje:**
  - **Bezpečné přihlášení** na vzdálený server
  - Přístup k **shellu** *(příkazovému řádku)*
  - **Bezpečný přenos souborů** *(pomocí scp, sftp)*
  - **Tunelování** dat *(port forwarding)*

**Zajišťuje:**

- **Autentizaci** *(ověření identity)*
- Integritu přenesených dat
- **Transparentní šifrování**
- **Volitelnou kompresi přenosu**

**Standard pro správu Linux/Unix serverů** na dálku

**FTP** *(File Transfer Protocol)*

- Protokol **pro přenos souborů mezi zařízeními v síti.**
- Pracuje na **aplikační vrstvě, využívá TCP/IP**, standardně porty:
  - 21 *(řídící spojení)*
  - 20 *(datové spojení)*
- Funguje **nezávisle na operačním systému**.
- Používá se pro:
  - **Upload/download souborů**
  - **Správu webového hostingu**
- **Nevýhoda: nezabezpečený** *(data a hesla se přenáší v otevřeném textu)*
- **Zabezpečené alternativy:** FTPS, SFTP *(součást SSH)*

# 12. Unix

## **DNS** *(Domain Name System)*

- **Systém doménových jmen** – převádí **čitelná doménová jména na IP adresy a naopak**.
- Pracuje na **aplikační vrstvě**, používá UDP port 53 *(v některých případech TCP)*.
- **Zajišťuje:**
  - Překlad např. **www.google.com → 142.250.74.68**
- **DNS je distribuovaná hierarchická databáze:**
  - Root servery
  - **TLD** servery *(.cz, .com, ...)*
  - **Autoritativní servery** *(např. správa domény)*
- Umožňuje **efektivní, rychlé a škálovatelné směrování požadavků v síti**.
- **IPv4: 32bitová adresa** *(např. 192.168.1.1)*, **IPv6: 128bitová** *(např. 2001:0db8::1)*

## **DHCP** *(Dynamic Host Configuration Protocol)*

- Protokol pro **automatické přidělování síťových parametrů nově připojeným zařízením**.
- Pracuje nad **transportní vrstvou TCP/IP**, využívá:
  - UDP porty 67 *(server)* a 68 *(klient)*
- **Automaticky přidělí klientovi:**
  - IP adresu
  - Masku podsítě
  - Výchozí bránu *(gateway)*
  - Adresy DNS serverů
- **IP adresa není přidělena trvale** – používá se pronájem *(lease)*.
- **Klient má běžící DHCP klient, který obnovuje** *(renewuje)* adresu před vypršením lease.
- **Eliminuje nutnost ručního nastavování IP adres**.

| Protokol | Vrstva | Port | Funkce | Zabezpečení |
|---|---|---|---|---|
| **Telnet** | Aplikační | 23 | Vzdálený přístup (shell) | ❌ Ne |
| **SSH** | Aplikační | 22 | Vzdálený přístup + šifrování | ✅ Ano |
| **FTP** | Aplikační | 20/21 | Přenos souborů | ❌ Ne (alternativy FTPS, SFTP) |
| **DNS** | Aplikační | 53 (UDP) | Převod domén/IP | 🔒 Ano (částečně – DNSSEC) |
| **DHCP** | Aplikační | 67/68 (UDP) | Automatická konfigurace | ✅ (v rámci sítě) |

# 12. Unix

## Virtualizace Unixu na MS Windows

- **Virtualizace PC** = technologie umožňující **spouštění více operačních systémů** *(OS)* **současně na jednom fyzickém počítači**.
- **V praxi:** na běžícím OS Windows lze spustit virtuální počítač *(VM)*, ve kterém běží např. **Unix nebo Linux**.

**Technické požadavky:**

1. **Podpora virtualizace procesorem a základní deskou**
   - Nutné: Intel VT-x nebo AMD-V (většina nových CPU je podporuje).
   - BIOS/UEFI: musí být zapnuta položka např. Intel Virtualization Technology (VT-x).
2. **Operační paměť** *(RAM)*
   - Minimálně 2 GB RAM pro plynulý chod hostitelského i virtuálního systému.
   - Doporučeno: 4 GB a více.
3. **Dostatek místa na disku**
   - Virtuální stroj vytváří velký soubor (virtuální disk), který zabírá několik GB.
   - Doporučeno: SSD disk pro rychlejší práci.

| Software | Licence | Popis |
|---|---|---|
| **VirtualBox** | Freeware (open source) | Nejoblíbenější volně dostupný nástroj. Podporuje mnoho OS, běží na Windows, Linux, macOS. |
| **Microsoft Virtual PC** | Freeware (ukončený vývoj) | Dříve řešení od Microsoftu, omezené jen na starší systémy Windows (např. XP Mode). |
| **VMware Workstation** | Komerční | Profesionální nástroj s širokou funkcionalitou, podporuje snapshoty, klonování, sdílené složky. |
| **Parallels Workstation** | Komerční | Původně zaměřen hlavně na virtualizaci Windows na macOS, existují i verze pro Windows. |

## Praktické využití virtualizace Unixu na Windows

- Testování a vývoj softwaru v Unixových systémech bez nutnosti instalace Linuxu na disk.
- Výuka Unixu/Linuxu ve školním prostředí na počítačích s Windows.
- Spuštění nástrojů, které jsou dostupné pouze v Unix/Linux prostředí (např. bash, gcc, make).

## Výhody virtualizace

- Neovlivňuje hlavní *(hostitelský)* OS.
- Lze rychle vytvářet, klonovat a mazat jednotlivé virtuální systémy.
- Virtuální systém je izolovaný – bezpečnější testování.
- Možnost vytvoření snapshotu *(uložení stavu VM)* pro návrat do konkrétního bodu.

**Nevýhody:**

- Vyšší hardwarové nároky *(RAM, CPU, disk)*.
- Menší výkon než u nativní instalace.
- Bez zapnuté virtualizační technologie v BIOS nelze spustit 64bitové OS.

# 12. Unix

## Skripty:

1. **Práce se signály v C (reakce na SIGINT):**

```c
#include <signal.h>
#include <stdio.h>
#include <unistd.h>

void ouch(int sig) {
    printf("Ouch! - Mám signál %d\n", sig);
    (void) signal(SIGINT, SIG_DFL); // Po zachycení nastaví zpět výchozí chování
}

int main() {
    (void) signal(SIGINT, ouch); // Nastaví handler na přerušení (Ctrl+C)
    while(1) {
        printf("Hello!\n");
        sleep(1);
    }
}
```

- **SIGINT =** signál přerušení *(např. Ctrl+C)*
- **Funkce signal()** nastavuje obslužnou funkci *(handler)* pro daný signál
- **SIG_DFL =** výchozí obsluha *(např. ukončení programu)*

**2.** Shellový skript

read B– načte vstup uživatele do proměnné B
if [ "$B" = "YES" ] – porovnání hodnoty
Příkazový blok then … else … fi je zakončen klíčovým slovem fi

```bash
#!/bin/bash

echo "start…."
read B

if [ "$B" = "YES" ]; then
    echo "dobry den"
else
    echo "dobry vecer"
fi

echo "END…."
```

## Správa uživatelů a skupin v Linuxu

- useradd [jméno] - Vytvoří nového uživatele
- userdel [jméno] - Smaže uživatele a volitelně i jeho domovský adresář (-r)
- usermod [volby] [jméno] - Modifikuje existujícího uživatele (např. změna shellu, skupiny)
- passwd [jméno] - Změní nebo nastaví heslo uživatele

## Skupiny

- groupadd [jméno] - Vytvoří novou skupinu
- groupdel [jméno] - Smaže skupinu
- groupmod [volby] [jméno] - Upraví název nebo GID skupiny

```
groupadd studenti          # vytvoření skupiny "studenti"
useradd st1 -g studenti    # vytvoření uživatele st1 a zařazení do skupiny studenti
chmod g+rx st1             # přidání práv pro skupinu k souboru/d. st1 (není jasné,
passwd st1                 # nastaví heslo pro uživatele st1
```

  - **-g** určuje primární skupinu uživatele při jeho vytvoření.
  - **chmod g+rx** = přidání práv read a execute skupině ke konkrétnímu objektu *(soubor/složka)*.
  - Při passwd je třeba zadat heslo **2×.**

# 12. Unix

## Konfigurace služeb v Unix/Linux

### Důležité systémové cesty

- **/etc/init.d/** nebo **/etc/rc.d/init.d/** - Obsahuje startovací skripty služeb (daemons)
- **/etc/** - Obsahuje konfigurační soubory systému a služeb
- **/etc/vsftpd/vsftpd.conf -** Hlavní konfigurační soubor pro FTP server vsftpd

### Spouštění a správa služeb *(SysV init)*

- Startovací skripty se nachází v:
  - **/etc/init.d/** *(Debian, Ubuntu)*
  - **/etc/rc.d/init.d/** *(Red Hat, CentOS, starší systémy)*

**Příklad práce se službou:**

- **/etc/init.d/vsftpd start** # Spustí FTP službu
- **/etc/init.d/vsftpd stop** # Zastaví službu
- **/etc/init.d/vsftpd reload** # Načte novou konfiguraci bez restartu

### chkconfig – správa automatického spouštění služeb

- **chkconfig --list -** Vypíše aktuální služby a jejich stav ve všech runlevelech
- **chkconfig --add [služba] -** Přidá službu pro správu chkconfig
- **chkconfig --del [služba] -** Odebere službu z chkconfig

### apropos – vyhledávání v manuálových stránkách

- **apropos ftp**
  - Vyhledá všechny příkazy, manuálové stránky nebo popisy obsahující slovo **"ftp"**
  - Užitečné při hledání příkazů podle funkce

### Daemon *(služba)*

- Daemon je **program běžící na pozadí**, bez přímé interakce s uživatelem.
- Odpojí se od terminálu *(klávesnice)*, často **startuje při bootu**.
- Typicky **nastavuje, sleduje nebo zpracovává síťové požadavky**.
- Např. **vsftpd, sshd, cron, httpd, mysqld,** aj.
- Internetový daemon – **XINETD**
  - Řídí **spouštění síťových služeb na vyžádání** *(on-demand)*.
  - Centrálně konfiguruje služby přes **/etc/xinetd.conf a /etc/xinetd.d/**

### Konfigurace FTP serveru (vsftpd)

- **/etc/vsftpd/vsftpd.conf**
- **Obsahuje parametry jako:**
  - anonymous_enable=YES/NO
  - local_enable=YES
  - write_enable=YES
  - chroot_local_user=YES
- **Spuštění služby:**
  - **/etc/init.d/vsftpd start** # Spustí
  - **/etc/init.d/vsftpd stop** # Zastaví
  - **/etc/init.d/vsftpd reload** # Znovu načte konfiguraci

# 12. Unix

## Základní příkazy:

| Příkaz / Název | Význam |
|---|---|
| **ls** | Výpis obsahu adresáře (listování) |
| **man** | Zobrazí manuálovou stránku příkazu |
| **q** | Ukončí zobrazení (man, more, less atd.) |
| **ls -all** | Zobrazí všechny soubory, včetně skrytých |
| **ls -al** | Dlouhý výpis souborů se všemi informacemi |
| **cd** | Přechod do domovského adresáře |
| **cd ..** | Přechod o úroveň výše |
| **pwd** | Zobrazí aktuální cestu (Print Working Directory) |
| **ls -l** | grep x0a |
| **/bin** | Obsahuje základní příkazy operačního systému |
| **/sbin** | Obsahuje systémové příkazy pro administrátora (roota) |
| **cd /** | Přechod do kořenového adresáře |
| **/dev** | Obsahuje zařízení a drivery (disky, porty apod.) |
| **/etc** | Konfigurační soubory a startovací skripty systému |
| **/mnt** | Slouží pro připojení externích zařízení (např. disků) |
| **/tmp** | Dočasný adresář |
| **/var** | Měnící se data – logy, maily, data serveru (např. FTP) |
| **mc -a** | Spustí Midnight Commander – textový správce souborů |
| **cat** | Vypíše celý obsah souboru |
| **more** | Výpis souboru po stránkách (nelze se vracet) |
| **less** | Výpis souboru s možností návratu a posunu v obou směrech |
| **mkdir** | Vytvoření nového adresáře |
| **cp** | Kopírování souborů nebo adresářů |
| **rm** | Mazání souborů (s -r i adresářů) |
| **ps** | Výpis běžících procesů uživatele |
| **sort** | Třídění obsahu (např. textových souborů) |