ÚVOD DO OPERAČNÍCH SYSTÉMŮ

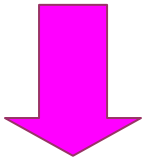
Ver.1.00

Operační systém (*Operating System*)

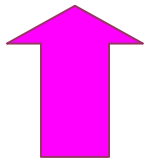
Definice, komponenty OS, vývoj a typy OS, služby OS, systémová volání, systémové programy, architektura

České vysoké učení technické Fakulta elektrotechnická

Studijní materiály a informace o předmětu



- http://measure.feld.cvut.cz/vyuka/predmety/bakalarske/navody
- http://aldebaran.feld.cvut.cz/vyuka/uvod_do_os



Použitá literatura

[1] Stallings, W.: Operating Systems. Internals and Design Principles. 4th Edition. Prentice Hall, New Jersey, 2001.

[2] Silberschatz, A. – Galvin, P. B. - Gagne, G. : Operating System Concepts. 6th Edition. John Wiley & Sons, 2003.

[3] Tanenbaum, A.: Modern Operating Systems. Modern Operating Systems. Prentice Hall, New Jersey, 2008.

[4] Firemní materiály IBM.

[5] Zděnek, J.: Úvod do operačních systémů 01 Architektura počítače. ČVUT FEL, Praha, 2010.

Operační systém (*Operating System*)

- **Operační systém** je software který:
  - řídí provádění uživatelských programů,
  - funguje jako rozhraní mezi aplikačním programem a fyzickými prostředky (resources) počítače,
  - spravuje všechny fyzické prostředky počítače,
  - vytváří lepší, jednodušší, přehlednější prostředí pro efektivní využití počítače (viz [3]).

- **Prostředky (*resources*) počítače**:
  - Procesory
  - Operační paměť (*Main Memory*)
  - Disky
  - Displej
  - Klávesnice
  - Myš
  - Síťové adaptéry
  - Další vstupní/výstupní zařízení (*Devices*)

Vrstvy počítačového systému

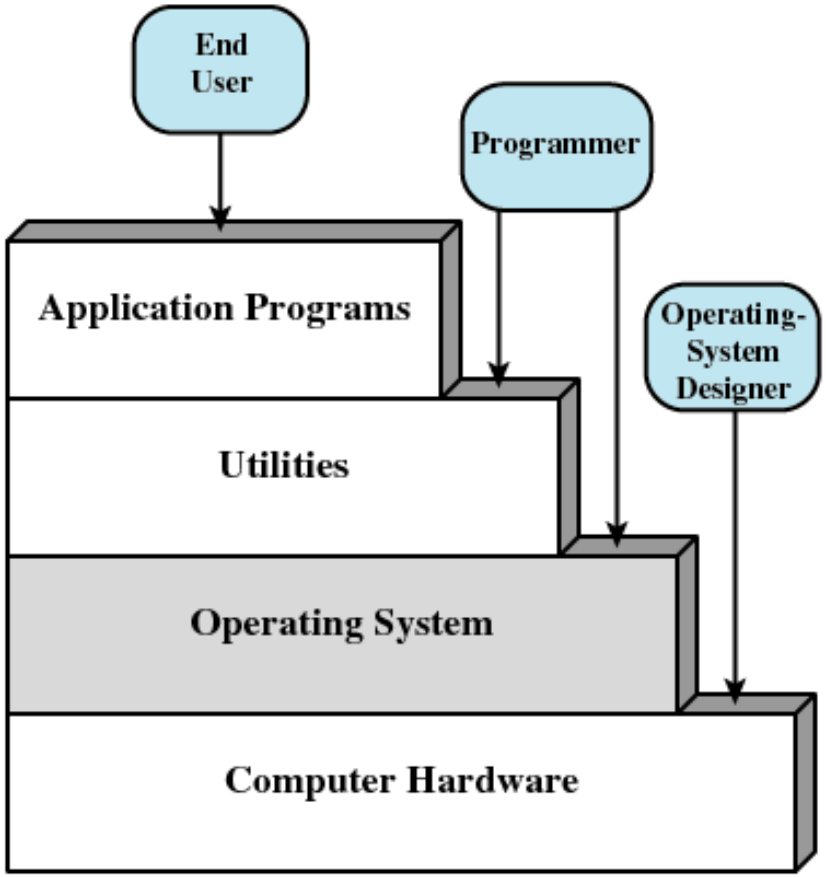
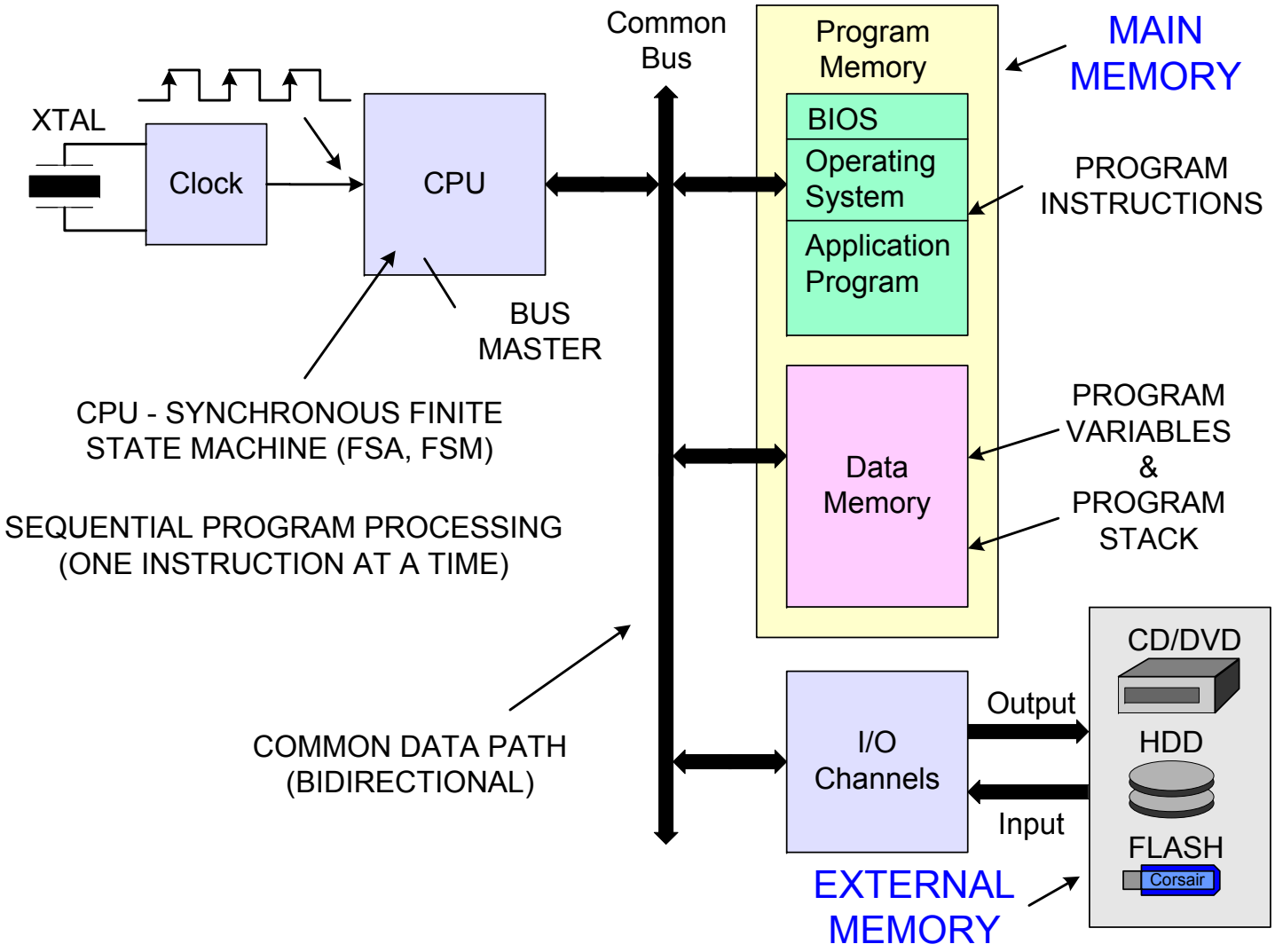


Figure 2.1 Layers and Views of a Computer System

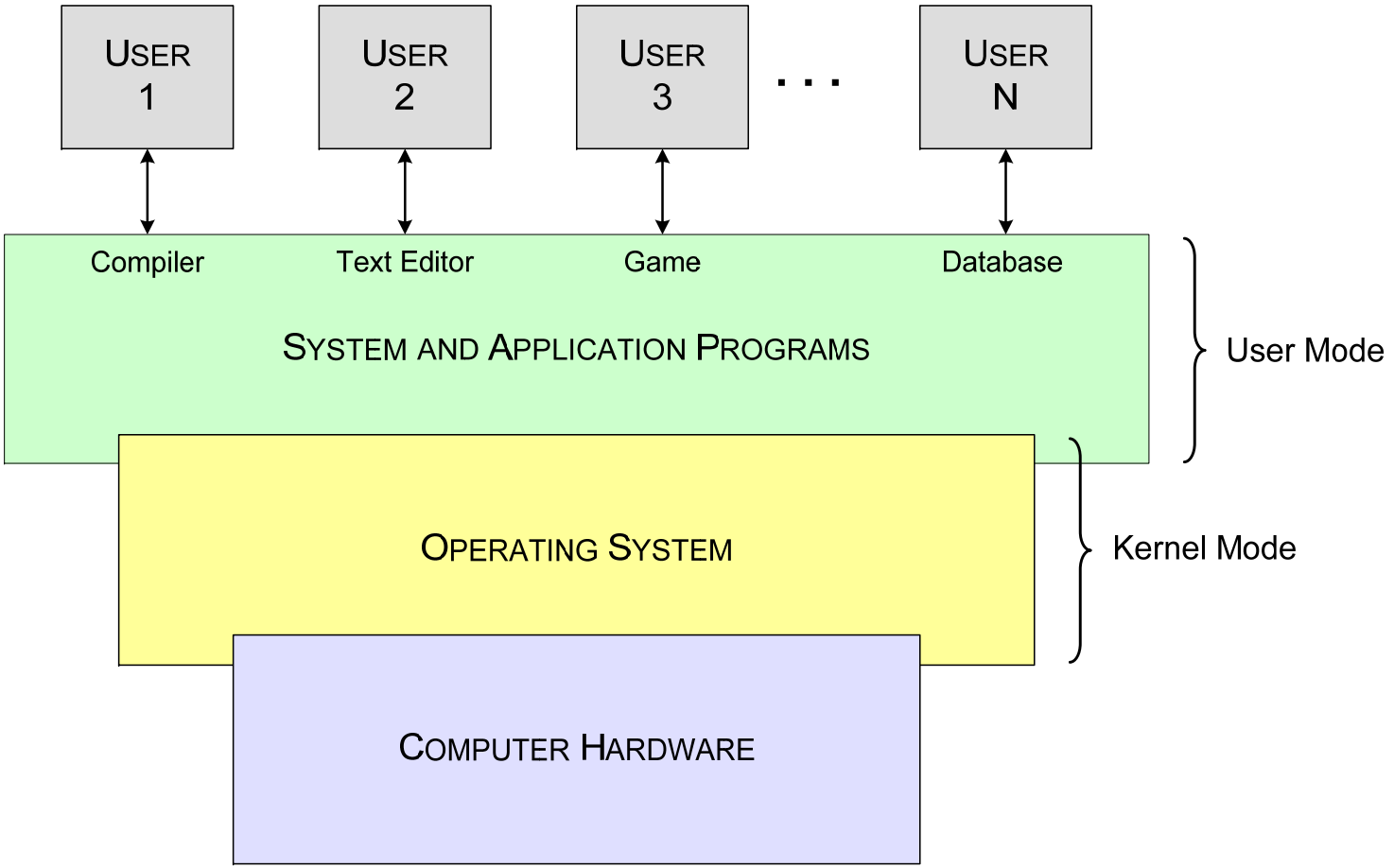
Viz [1]

OS jako **Správce prostředků** *(Resource Manager)*



Viz [5]

Komponenty počítačového systému

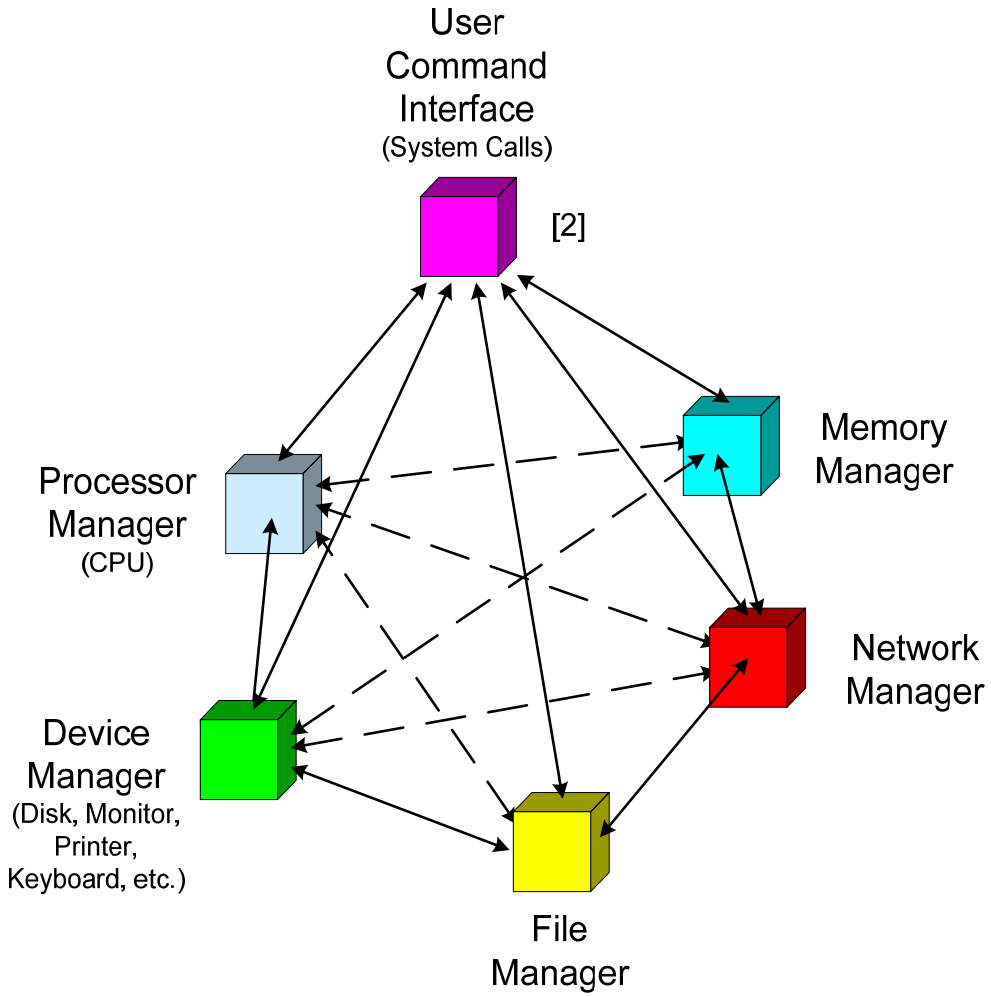


Viz [5]

Komponenty operačního systému

- Správa procesů (*Process Management*)
- Správa operační paměti (*Main-Memory Management*)
- Správa souborů (*File Management*)
- Správa vstupů/výstupů (*I/O System Management*)
- Správa sekundární paměti (*Secondary-Storage Management*)
- Správa síťových služeb (*Networking*)
- Systém zabezpečení (*Protection System*)
- Příkazový interpeter (*Command-Interpreter System*)

Komponenty operačního systému



Viz [5]

Vývoj operačních systémů – „sériové zpracování“

Historie

Pět vývojových generací počítačů a počítačových systémů detailně popsáno v přednášce Úvod do operačních systémů 01 Architektura počítače (viz [5])

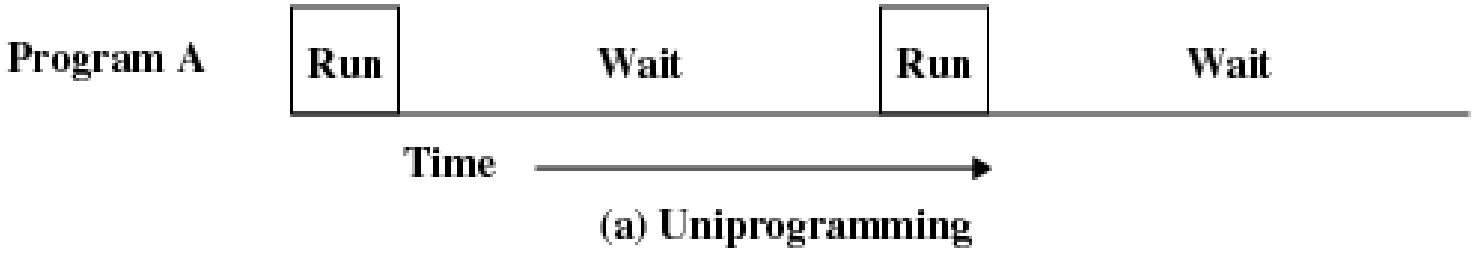
Historický vývoj OS

Sériové zpracování (*Serial Processing*)

- Operační systém neexistoval
- Jednotlivé operace spouštěl programátor včetně zadávání vstupních dat
- Později byly zavedeny vstupy a výstupy pomocí děrných štítků

Vývoj OS – jednoúlohové dávkové systémy *(Simple Batch Systems)*

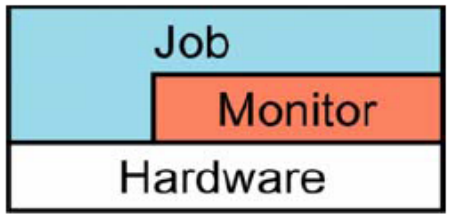
Jednoúlohový dávkový systém – umožnil zpracování programu bez přítomnosti programátora. Určen pro vědecké výpočty a komerční zpracování dat. Program a data umístěny na děrných štítcích (příp. páskách).



Monitor – jednoduchý program pro ovládání a monitoring počítače.

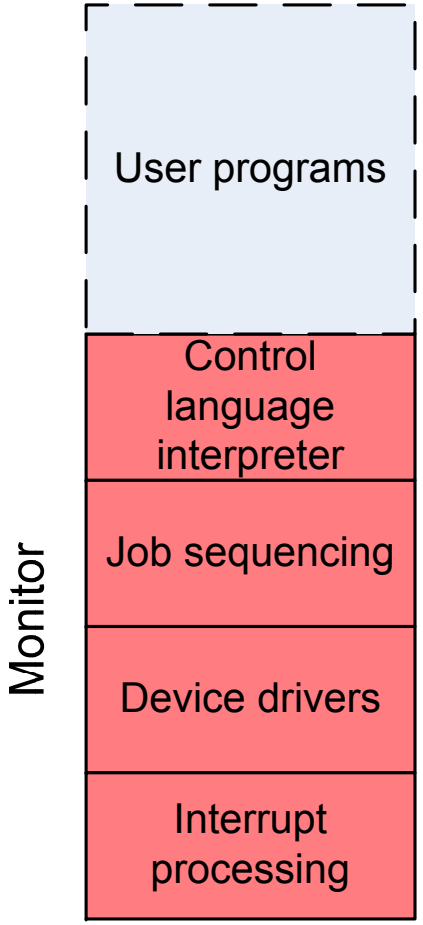
Viz [1]

Vývoj OS – jednoúlohové dávkové systémy



Monitor – umožňoval:

- zavedení a spuštění programu (*job*)
- jednoduché ladění
- monitorování činnosti počítače
- dávkové zpracování



Vývoj OS – jednoúlohové dávkové systémy

Job Control Language (JCL):

- Speciální programovací jazyk pro ovládání monitoru
- Příkazy
  - Identifikace nové úlohy ($ JOB)
  - Specifikace kompilátoru
  - Překlad programu, umístění do paměti, spuštění („compile, load and go“)
  - Načítání dat
  - Ukončení úlohy

```
$ JOB
$ FORT
 . . .
< source program cards >
 . . .
$ LOAD
$ RUN
 . . .
< data cards >
 . . .
$ END
```

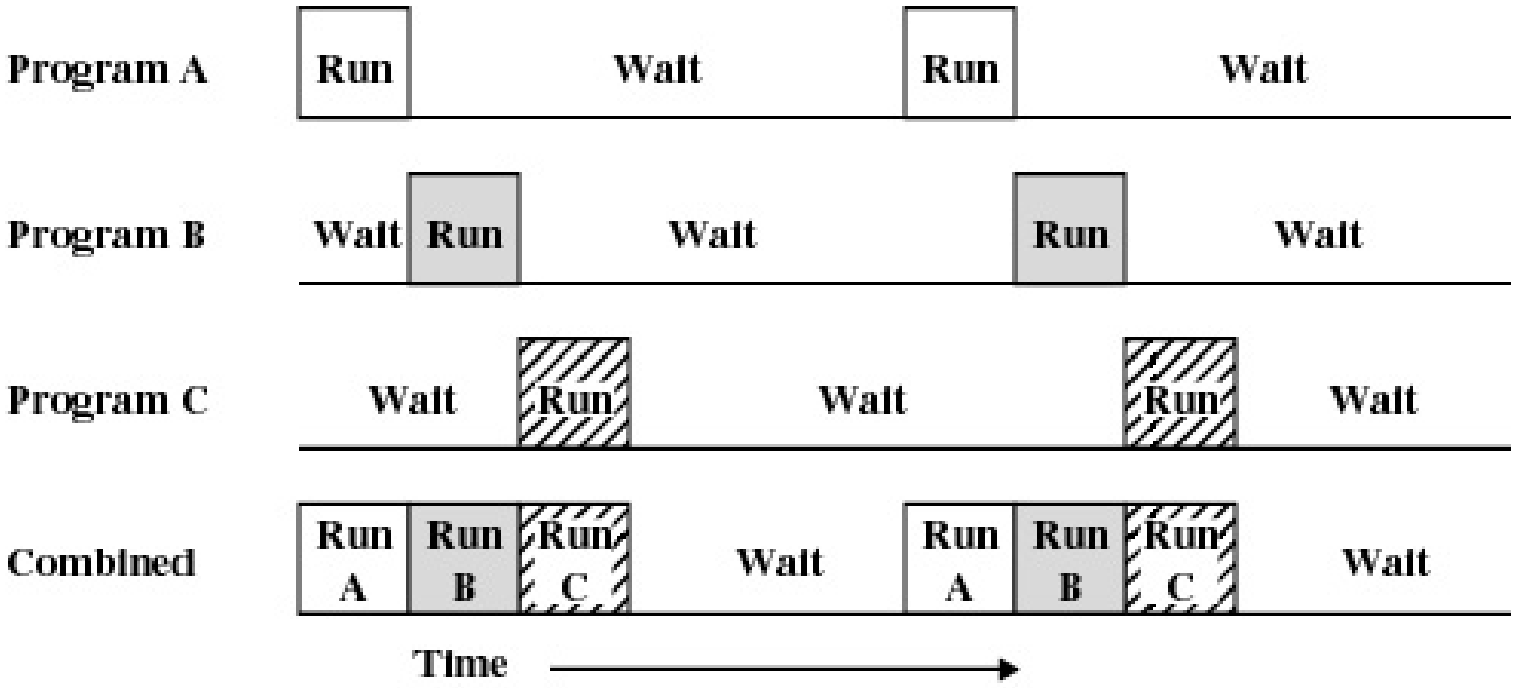
Vývoj OS – víceúlohové dávkové systémy (*Multiprogrammed Batch Systems*)

Víceúlohový dávkový systém – umožnil výrazně vyšší využití procesoru.

Dávkové multiprogramování (*batch multiprogramming, multitasking*)

- v paměti počítače je současně více úloh
- pokud probíhá V/V operace, je procesoru prostřednictvím OS předána další úloha

Vývoj OS – víceúlohové dávkové systémy



(c) Multiprogramming with three programs

Viz [1]

Vývoj OS – víceúlohové dávkové systémy

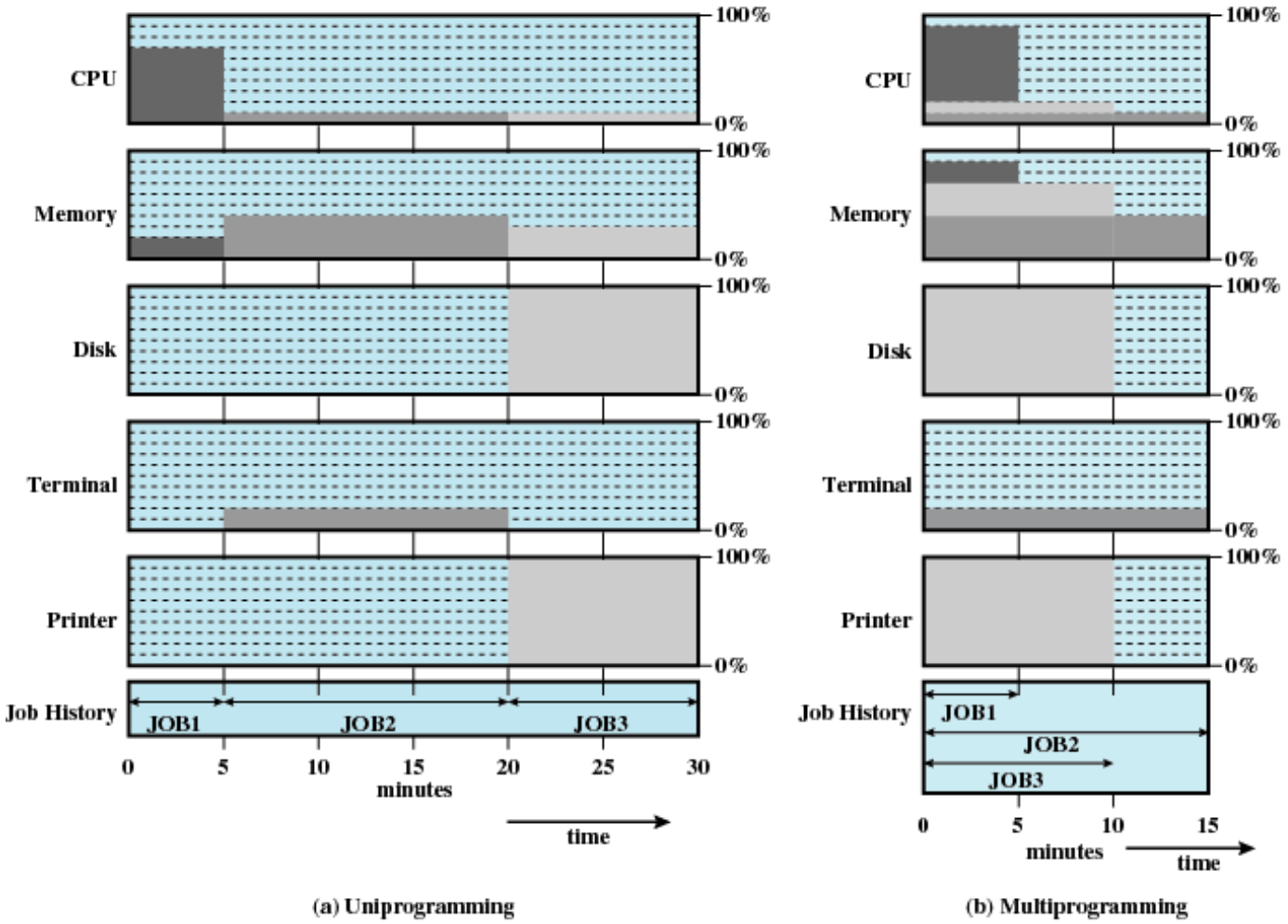


Figure 2.6 Utilization Histograms

Viz [1]

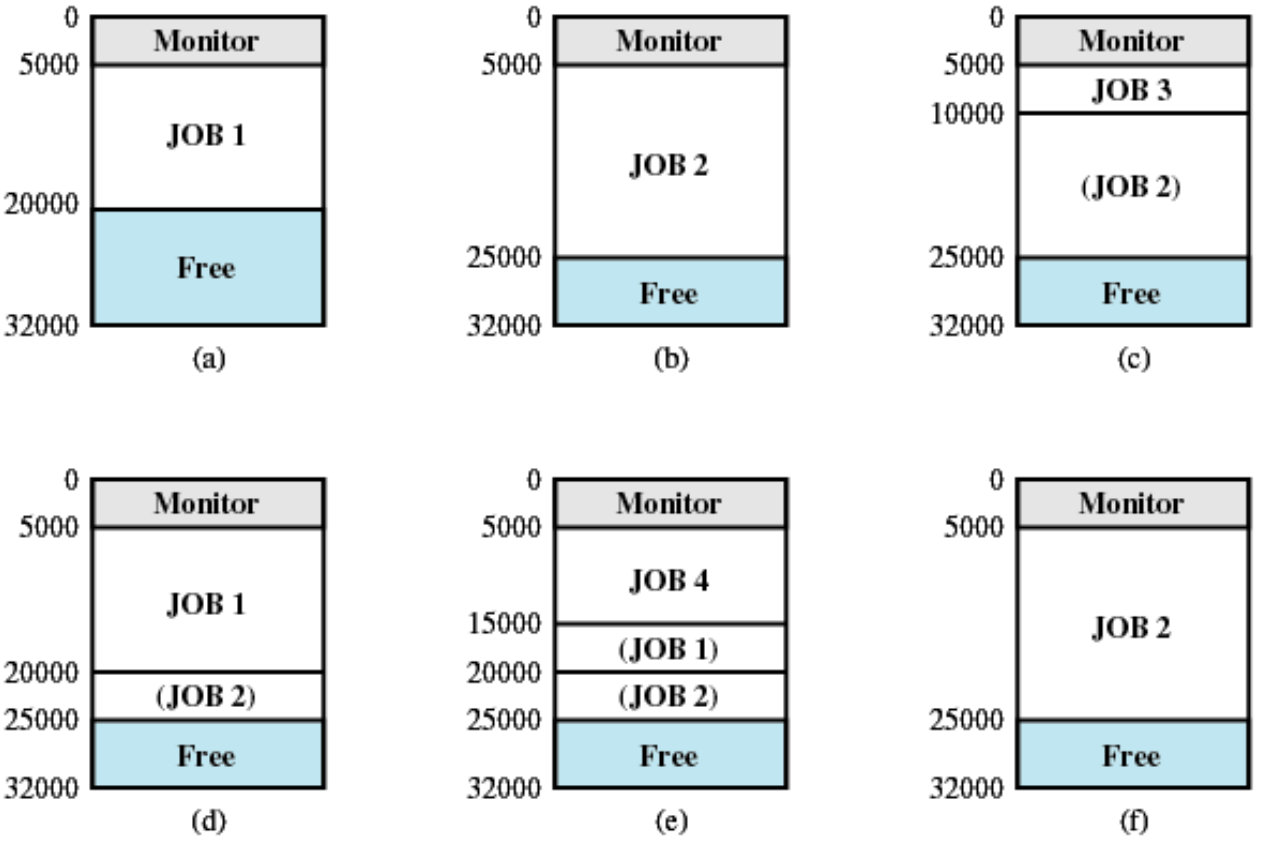
Vývoj OS – víceúlohové systémy se sdílením času (*Time-Sharing Systems*)

Víceúlohový systémy se sdílením času – umožnily využití počítače více uživateli.

Time-sharing (Time-slicing)

- úloha je procesoru přidělována na krátký časový interval (*time slice, time quantum*) - řádově jednotky až desítky milisekund
- sdílení času je optimalizováno na dosažení minimálního času odezvy

Vývoj OS – víceúlohové systémy se sdílením času



CTSS
Compatible Time-Sharing System

Figure 2.7 CTSS Operation

Viz [1]

Služby operačního systému

OS zajišťuje prostředí pro běh (=vykonávání) programů prostřednictvím služeb.

OS poskytuje:

- služby pro programy,
- služby pro uživatele.

Základní služby OS:

- provádění programů (*program execution*)
- vstupně/výstupní operace (*I/O operation*)
- manipulace se soubory (*file system manipulation*)
- komunikace mezi procesy (*process communication*)
- detekce chyb (*error detection*)

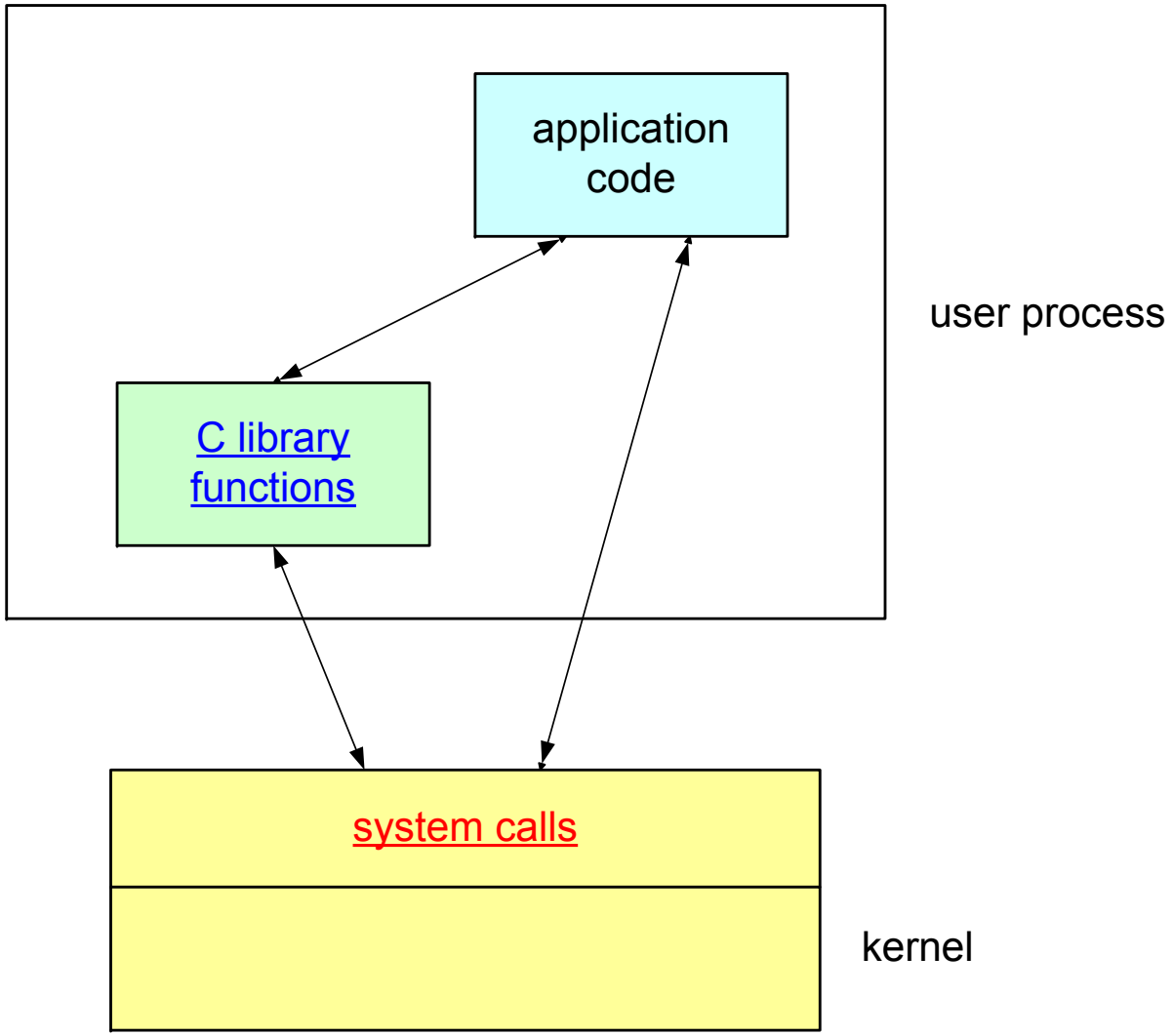
Systémová volání (*System Calls*)

Systémová volání (=volání jádra) představují rozhraní mezi procesy a jádrem operačního systému.

Typy volání jádra:

- Řízení procesů (*Process Control*)
  - load, execute
  - end, abort
  - create process, terminate process
- Správa souborů (*File Management*)
  - create, delete file
  - open, close
  - read, write
- Správa vstupních/výstupních zařízení (*Device Management*)
- Komunikace (*Communication*)
- Udržování informací (*Information Maintenance*)

Systémová volání



Systémová volání jako API OS

```
int main(...)
{
        ...
        if ((pid = fork()) == 0)                    // vytvoření nového procesu
        {
                fprintf(stdout, "Child pid: %i\n", getpid());
                err = execvp(command, arguments);   // toto je nový proces („dítě“)
                fprintf(stderr, "Child error: %i\n", errno);
                exit(err);
        }
        else if (pid > 0)                           // toto je rodičovský proces
        {
                fprintf(stdout, "Parent pid: %i\n", getpid());
                pid2 = waitpid(pid, &status, 0);    // čekání na ukončení procesu
        }
        ...
        return 0;
}
```

Příklad: implementace příkazu shellu v Unixu pomocí volání jádra a knihovních funkcí jazyka C

Systémové programy (*System Programs*)

Systémové programy zajišťují prostředí pro vývoj a provádění programů.

Typy systémových programů:

- Manipulace se soubory (*file manipulation*)
  - manipulace se soubory a adresáři
  - úprava obsahu souborů, textové editory
- Stavové informace (*status information*)
  - zjišťování systémových informací
- Podpora programovacích jazyků (*programming-language support*)
  - vývoj, ladění a spouštění programů
- Komunikace (*Communication*)
  - komunikace s procesy, uživateli, vzdálenými počítačovými systémy

Typy operačních systémů (podle využití v praxi)

- OS pro střediskové počítače (*Mainframes*)
  - dávkové zpracování, zpracování transakcí, timesharing
  - příklady: OS/390, OS/360 (IBM)
- OS pro servery
  - příklady: UNIX, Windows, Linux, VMS
- OS pro osobní počítače
  - příklady: Windows 95/98/ME, Windows 2000/XP/VISTA/Win2007, Macintosh OS,Linux.
- Paralelní systémy
  - Multiprocesorové systémy pro speciální účely
  - *clusters*

Typy operačních systémů (podle využití v praxi)

- Distribuované systémy
  - Využívají vhodné sítě LAN, WAN, aj.
  - Architektury klient-server, *peer-to-peer*
- RTOS (Real Time OS)
  - Speciální aplikace (průmysl, doprava, vědecké experimenty, „vestavěné“ aplikace)
  - *Hard/soft RTOS*
  - příklady: VxWorks, RT Linux, RTX, PharLap, … (a většina *Embedded OS*)
- Vestavěné OS (Embedded OS)
  - OS pro komerční i nekomerční aplikace (TV přijímače, mobilní telefony, pračky, digitální fotoaparáty, aj.)
  - OS pro speciální paměťové karty, PDA, aj.
  - většinou mají vlastnosti RTOS
  - příklady: uCLinux, FreeRTOS, Android, QNX, Symbian

Architektura OS

- Monolitické systémy (*Monolithic Systems*)
  - OS běží jako jediný program v režimu jádra (*kernel mode*)
  - Systém je napsán jako soubor procedur sestavených do jediného spustitelného binárního programu
  - Dosahuje se vysoké efektivity (částečně na úkor robustnosti?)
  - Často se používá modulární návrh
  - Typické příklady: Linux, Solaris, HP-UX, MS-DOS, Windows (částečně), Mac OS (do verze 8.6), …

Architektura OS

- Vrstevné systémy (*Layered Systems*)
  - OS je hierarchicky rozdělen do vrstev
  - Příklady: THE system – autor E.W.Dijkstra, MULTICS

- Systémy založené na mikrojádře (*Microkernel Architecture*)
  - Systém nemá monolitické jádro
  - Existuje pouze „mikrojádro“ se základními funkcemi (správa paměti, procesů, souborů, plánování, IPC, zpracování přerušení) spuštěné v „*kernel mode*“
  - Vše ostatní běží v „*user mode*“
  - Příklady: PikeOS, QNX, Minix 3, Symbian

Architektura OS

Porovnání vrstevného systému a systému založeném na mikrojádře

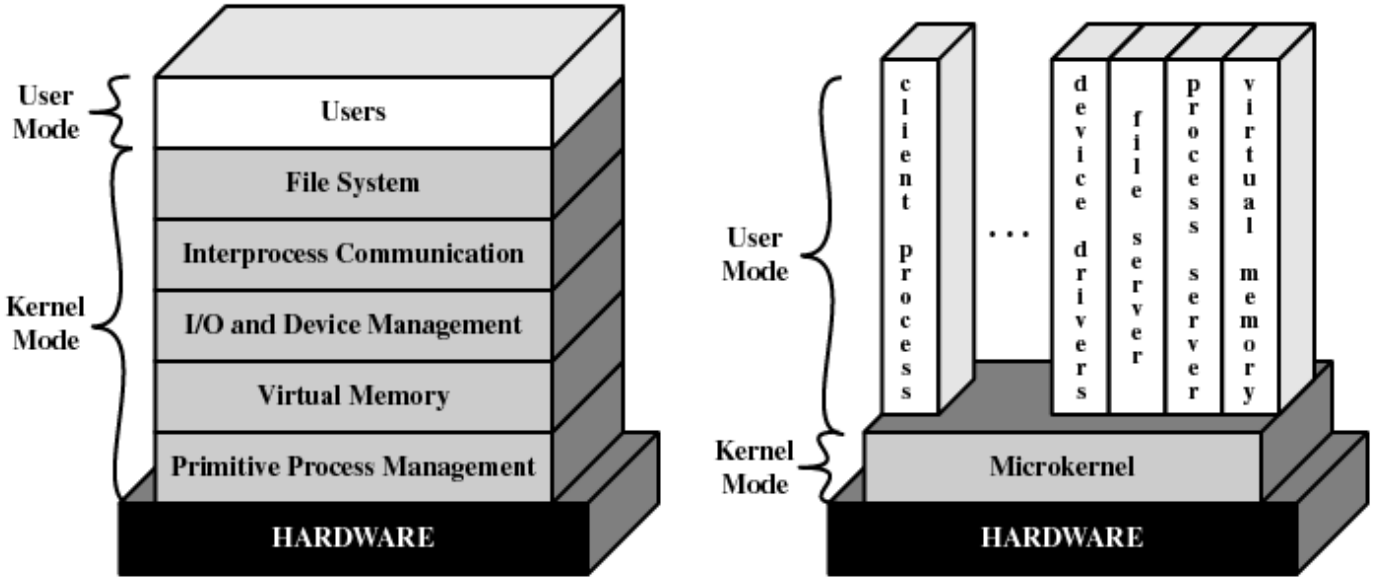
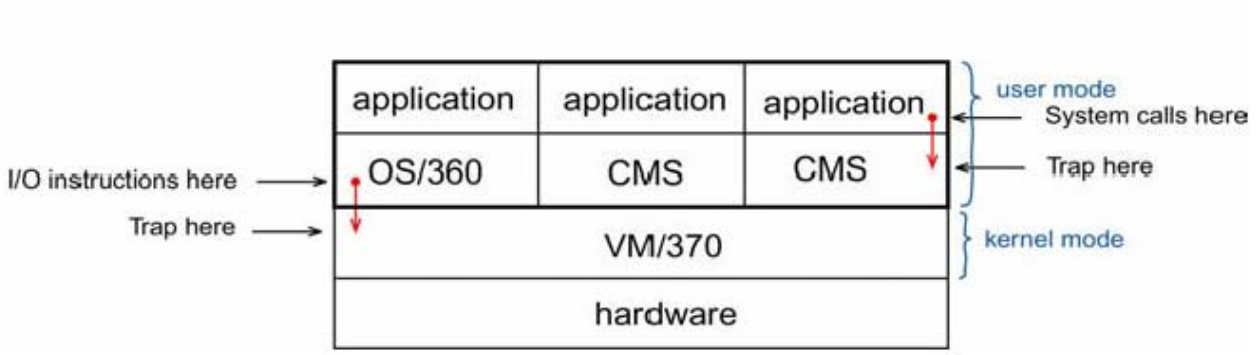


Figure 4.10 Kernel Architecture

Viz [1]

Architektura OS

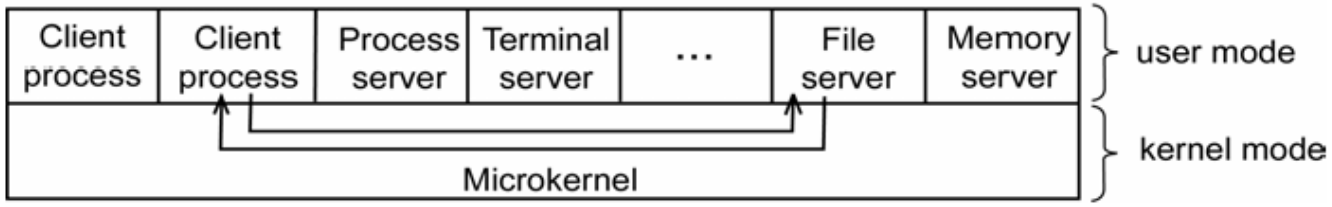
- Virtuální stroje (*Virtual Machines*)
  - Virtuální monitor (*Virtual Machine Monitor*) běží jako jediný program v režimu jádra (*kernel mode*) těsně nad vrstvou HW
  - VM poskytuje „uživatelské“ vrstvě plnohodnotných virtuálních strojů (včetně *kernel/user mode*, *I/O, interrupts,…)*
  - Typické příklady: OS/360



Viz [4]

Architektura OS

- Model klient-server (*Client-Server Model*)
  - Jádro je minimalizováno, využívá se *microkernel*
  - Proces klient posílá požadavky, proces server je vykoná a vrací výsledek
  - *Microkernel* řídí přepínání kontextu a výměnu zpráv mezi klientem a serverem
  - Lze implementovat v podobě distribuovaného systému



Tradiční architektura OS Unix

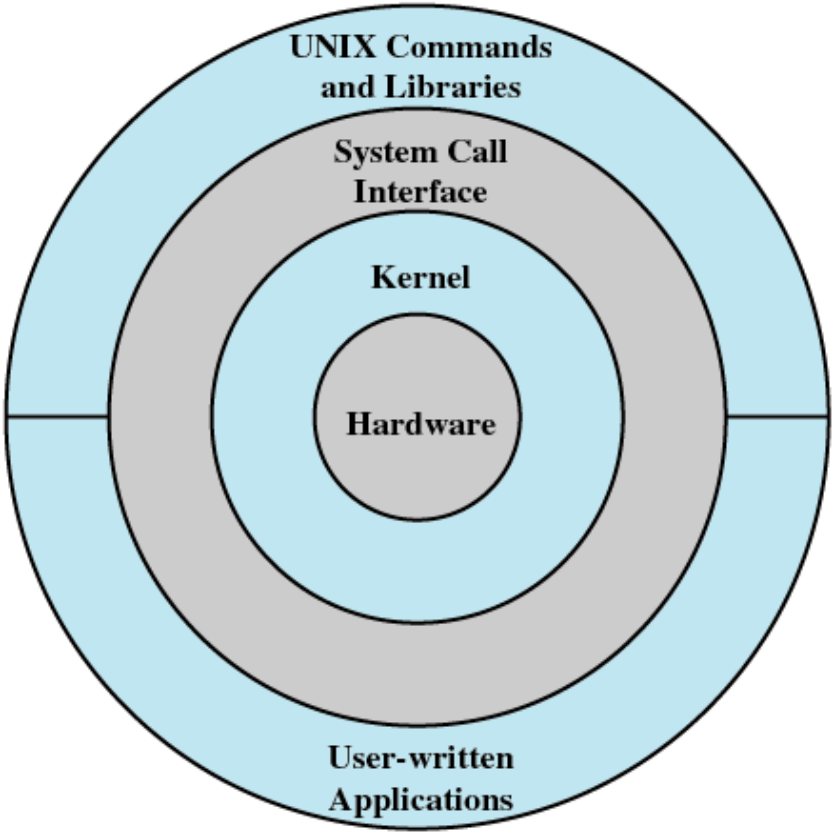


Figure 2.14 General UNIX Architecture

Viz [1]

Tradiční architektura jádra OS Unix

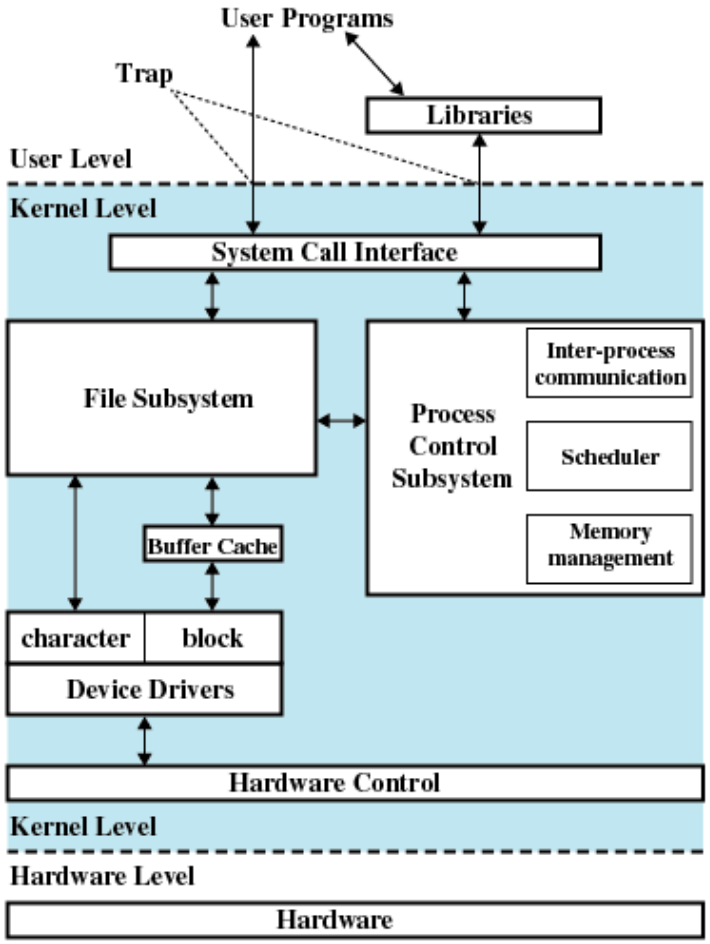


Figure 2.15 Traditional UNIX Kernel [BACH86]

Viz [1]

Moderní architektura jádra OS Unix

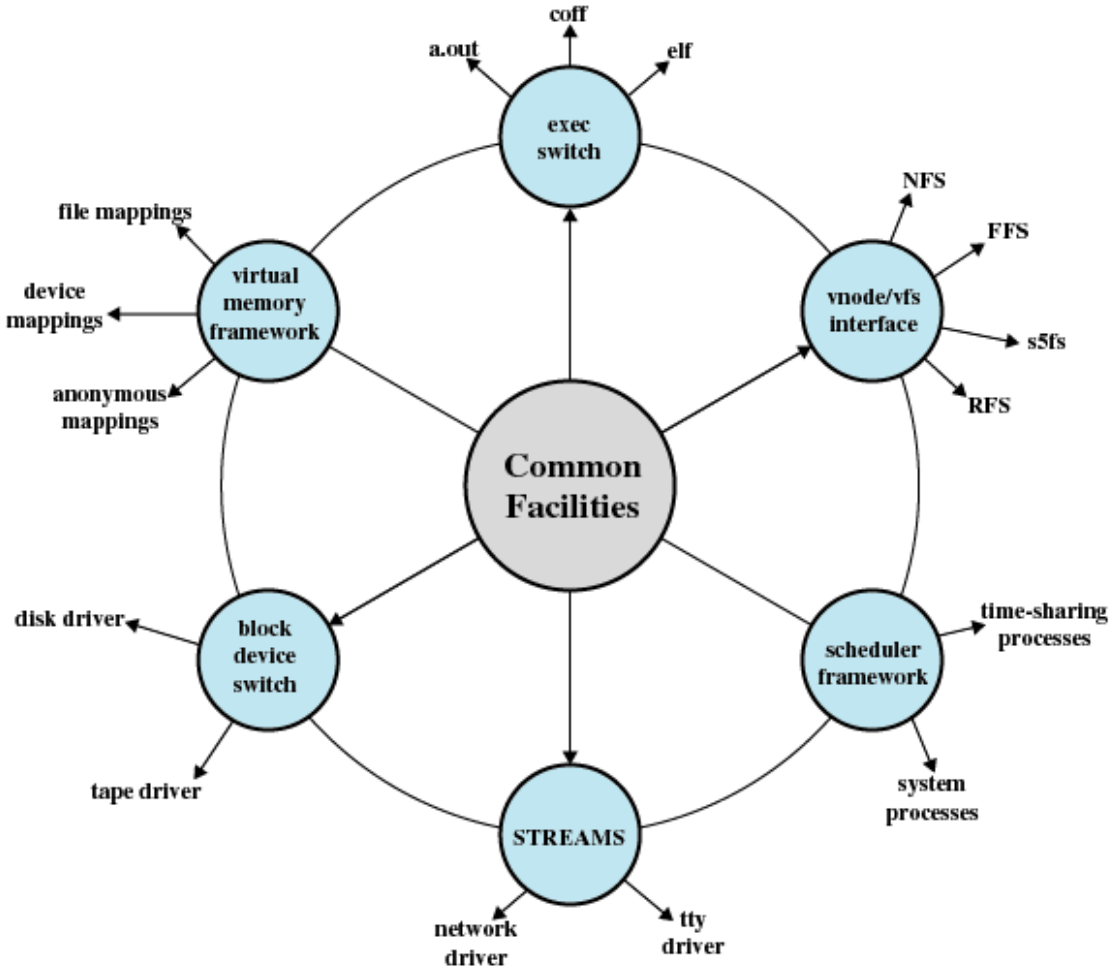


Figure 2.16 Modern UNIX Kernel [VAHA96]

Viz [1]

ÚVOD DO OPERAČNÍCH SYSTÉMŮ

KONEC 2. přednášky

České vysoké učení technické Fakulta elektrotechnická