Operační systémy

IOS 2024/2025

Garant předmětu: **Adam Rogalewicz**
rogalew@fit.vut.cz
Autor slidů: **Tomáš Vojnar**
vojnar@fit.vut.cz

Vysoké učení technické v Brně
Fakulta informačních technologií
Božetěchova 2, 612 66 Brno

Motivace

❖ Operační systém je významnou částí prakticky všech výpočetních systémů, jež typicky zahrnují:

- hardware,
- operační systém (OS),
- uživatelské aplikační programy a
- uživatele.

❖ I pro ty, kteří se nebudou podílet na vývoji žádného OS, má studium OS velký význam:

- OS mají významný vliv na fungování celého výpočetního systému a znalost principů jejich fungování je tedy zapotřebí při vývoji:
  - hardware, na kterém má běžet nějaký OS a
  - software, který má prostřednictvím nějakého OS efektivně využívat zdroje hardware, na kterém běží – což by mělo platit vždy.

- OS jsou obvykle značně komplikované systémy, jež prošly a procházejí dlouhým a nákladným vývojem, při kterém:
  - byla učiněna řada rozhodnutí ovlivňujících filozofii vývoje, chování vývojářů, trendy a trh IT: otevřený x uzavřený software, vývoj SW v komunitách vývojářů, ovlivňování trhu IT na základě vlastností OS, ...
  - byla použita řada zajímavých algoritmů, způsobů návrhu architektury SW, metodologií a technik softwarového inženýrství apod., jež jsou inspirující i mimo oblast OS.

❖ Zapojení se do vývoje některého OS či jeho části není navíc zcela nepravděpodobné:

- Není např. možné dodávat nově vyvinutý hardware (rozšiřující karty apod.) bez podpory v podobě ovladačů (driverů).
- Pro potřeby různých aplikací (např. u vestavěných počítačových systémů) může být zapotřebí vyvinout/přizpůsobit vhodný OS.
- Zapojení se do vývoje některého otevřeného OS (Linux, FreeBSD, mikrojádra typu L4, ...) je značnou intelektuální výzvou a může přinést příslušné intelektuální uspokojení.

Organizace studia

❖ Přednáška:

- 3h/týden (případné bonusové body za účast),
- principy a vlastnosti operačních systémů (UNIX obvykle jako případová studie),
- UNIX z uživatelského a částečně programátorského pohledu.

❖ Samostatná práce:

- min. 2h/týden,
- samostudium, experimenty, ..., úkoly (+ dobrovolná demo-cvičení).

❖ Hodnocení:

projekty	30b	
půlsem. zkouška	10b	(zápočet: min 15b z půlsem. zk. a projektů)
semestrální zkouška	60b	(minimum: 27b)
celkem	100b	

Zdroje informací

- Materiály z přednášek a odkazy na externí zdroje:
`http://www.fit.vut.cz/study/courses/IOS/private/`

- Aktuality, diskusní fóra kursu a informace k pro: E-learningu VUT – diskutujte!

- Literatura: koupit či vypůjčit (např. v knihovně FIT, ale i v jiných knihovnách).
  - Je-li některá kniha dlouhodobě vypůjčena některému zaměstnanci, neváhejte a kontaktujte ho.
- Internet:
  - vyhledávače (google, ...); často jsou užitečné dokumenty typu HOWTO, FAQ, ...
  - encyklopedie: `http://www.wikipedia.org/` – velmi vhodné při prvotním ověřování některých bodů z přednášek při studiu, lze pokračovat uvedenými odkazy, nutné křížové ověřování.
  - dokumentační projekty: `http://www.tldp.org/`, ...
  - ChatGPT (a podobné nástroje): Rychlá orientace a nápověda. Nalezení vhodných knihovních funkcí, příkazů shellu, jejich parametrů a příkladů použití. Nutné křížové ověřování, tyto nástroje mohou začít halucinovat.

- UNIX, Linux: program `man` („RTFM!“), GNU `info` a další dokumentace (`/usr/share/doc`, `/usr/local/share/doc`), ...

Literatura

[1] Silberschatz, A., Galvin, P.B., Gagne, G.: Operating Systems Concepts, 10th ed., John Wiley & Sons, 2018. (Významná část přednášek je založena na tomto textu.)

[2] Tanenbaum, A.S.: Modern Operating Systems, 4th ed., Pearson, 2014.

[3] Tanenbaum, A.S., Woodhull, A.S.: Operating Systems Design and Implementation, 3rd ed., Prentice Hall, 2006.

[4] Raymond, E.S.: The Art Of Unix Programming, Addison-Wesley, 2003.
`http://www.catb.org/~esr/writings/taoup/html/`

[5] Yosifovich, P., Ionescu, A. Russinovich, M., Solomon, D.: Windows Internals, 7th ed., Microsoft Press, 2017.

[6] Skočovský, L.: Principy a problémy operačního systému UNIX, 2. vydání, 2008.
`http://skocovsky.cz/paposu2008/`

Uvedeny jsou pouze některé základní, zejména přehledové knihy. Existuje samozřejmě řada dalších knih, a to včetně knih specializovaných na detaily různých subsystémů operačních systémů (plánovač, správa paměti, souborové systémy, ovladače, ...). Tyto knihy často vycházejí přímo ze zdrojových kódů příslušných subsystémů, které uvádějí s příslušným komentářem.

Základní pojmy

Význam pojmu operační systém

❖ Operační systém je program, resp. kolekce programů, která vytváří spojující mezivrstvu mezi hardware výpočetního systému (jenž může být virtualizován) a uživateli a jejich uživatelskými aplikačními programy.

❖ Cíle OS – kombinace dvou základních, do jisté míry protichůdných cílů, jejichž poměr se volí dle situace:

- Maximální využití zdrojů počítače:
  - drahé počítače, levnější pracovní síla,
  - zejména dříve, dnes na speciálních architekturách (superpočítače, ...)
- Jednoduchost použití počítačů:
  - levné počítače a drahá pracovní síla,
  - dnes převažuje.

❖ Dvě základní role OS:

- Správce prostředků (paměť, procesor, periferie).
  - Dovoluje sdílet prostředky efektivně a bezpečně.
  - Více procesů sdílí procesor, více procesů sdílí paměť, více uživatelů a souborů obsazuje diskový prostor, ...

- Tvůrce prostředí pro uživatele a jejich aplikační programy (tzv. *virtuálního počítače*).
  - Poskytuje standardní rozhraní, které zjednodušuje přenositelnost aplikací a zaučení uživatelů.
  - Poskytuje abstrakce:
    - Technické vybavení je složité – práci s ním je nutné zjednodušit.
    - Příklady abstrakcí: proces[a], soubor[b], virtuální paměť, ...
    - Problémy abstrakcí: menší efektivita, nepřístupné některé nízkoúrovňové operace.

---

[a] *Program: předpis, návod na nějakou činnost zakódovaný vhodným způsobem (zdrojový text, binární program). Proces: činnost řízená programem.*

[b] *Soubor: kolekce záznamů (typicky prostě bajtů) sloužící (primárně) jako základní jednotka pro ukládání dat na vnějších paměťových médiích. Adresář: kolekce souborů.*

❖ Výše uvedené není zadarmo! OS spotřebovává zdroje (paměť, čas procesoru).

❖ OS se obvykle chápe tak, že zahrnuje:

- jádro (kernel),
- systémové knihovny a utility (systémové aplikační programy),
- textové a/nebo grafické uživatelské rozhraní.

❖ Přesná definice, co vše OS zahrnuje, však neexistuje:

- Někdy bývá OS ztotožňován takřka pouze s jádrem.
- GNU – *GNU is Not UNIX*: Projekt vývoje „svobodného“ (*free*) OS, který zahrnuje jádro (Linux, Hurd), utility, grafické i textové rozhraní (bash, Gnome, KDE), vývojové prostředky a knihovny (gcc, gdb, ...) i řadu dalších programů (včetně kancelářských programů a her).
- Je či není prohlížeč Internetu/přehrávač videa/... nedílnou součástí OS?
- Distribuce Linux/Windows/MacOS jako OS.

Jádro OS

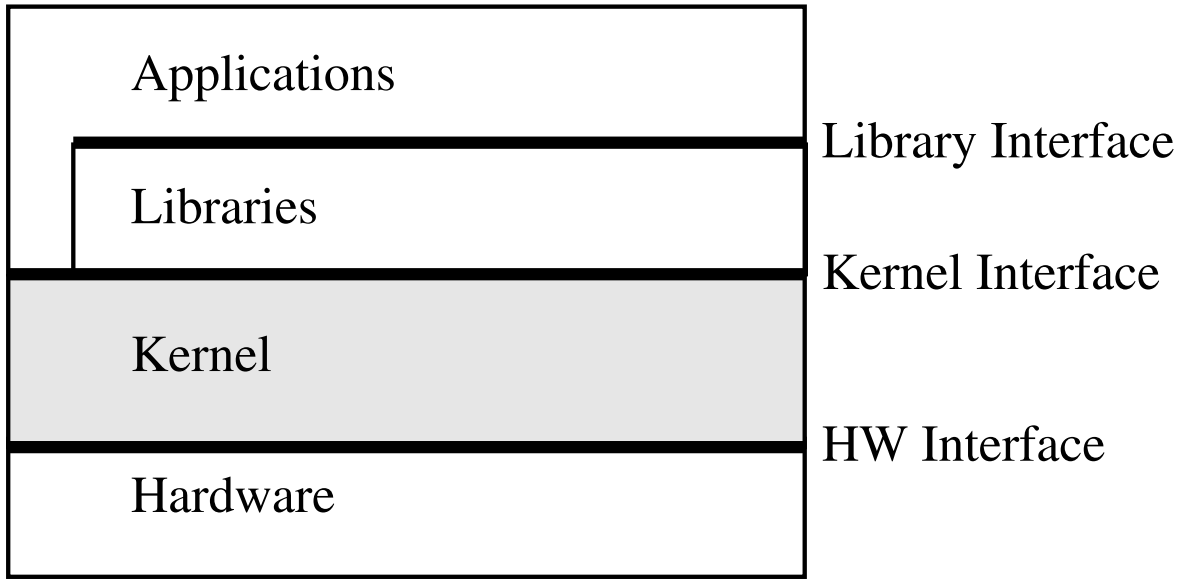
❖ Jádro OS je nejnižší a nejzákladnější část OS.

- Zavádí se první a běží po celou dobu běhu počítačového systému:
  - tzv. *reaktivní* spíše než *transformační* program.
- Navazuje přímo na hardware (příp. virtualizovaný HW) a typicky ho pro uživatele a uživatelské aplikace zcela zapouzdřuje.
- Běží obvykle v privilegovaném režimu.
  - V tomto režimu může provádět libovolné operace nad (virtualizovaným) HW počítače.
  - Za účelem oddělení a ochrany uživatelů a jejich aplikačních programů nesmí tyto mít možnost do tohoto režimu libovolně vstupovat.
  - Nutná HW podpora v CPU.
- Zajišťuje základní správu prostředků a tvorbu prostředí pro vyšší vrstvy OS a uživatelské aplikace.
  - To, jaké konkrétní služby jádro nabízí, je výsledkem zvoleného kompromisu mezi efektivitou, bezpečností, flexibilitou a příp. dalšími aspekty.
  - Jedná se ovšem minimálně o tu část služeb, která bezprostředně vyžaduje interakci s HW (přepínání kontextu, zavádění stránek, nastavování parametrů HW apod.).

❖ Systémové i uživatelské aplikační programy mohou explicitně žádat jádro o služby prostřednictvím systémových volání (*system call*). Děje se tak přímo nebo nepřímo s využitím specializovaných instrukcí (např. u x86 softwarové přerušení nebo `SYSCALL`/`SYSENTER`), které způsobí kontrolovaný přechod do režimu jádra.

❖ Rozlišujeme dva typy rozhraní OS:

- Kernel Interface: přímé volání jádra specializovanou instrukcí (případně jen zapouzdřenou v obálce ve vyšším programovacím jazyce)
- Library Interface: volání funkcí ze systémových knihoven, které mohou (ale nemusí) vést na volání služeb jádra



Typy jader OS

❖ Monolitická jádra:

- Vytváří vysokoúrovňové komplexní rozhraní s řadou služeb a abstrakcí nabízených vyšším vrstvám.
- Všechny subsystémy implementující tyto služby (správa paměti, plánování, meziprocesová komunikace, souborové systémy, podpora síťové komunikace apod.) běží v privilegovaném režimu a jsou těsně provázané za účelem vysoké efektivity.

❖ Vylepšením koncepce monolitických jader jsou monolitická jádra s modulární strukturou:

- Umožňuje zavádět/odstraňovat subsystémy jádra v podobě tzv. modulů za běhu.
- Např. FreeBSD či Linux (`lsmod`, `modprobe`, `rmmod`, ...)

❖ Mikrojádra:

- Minimalizují rozsah jádra a nabízí jednoduché rozhraní, s jednoduchými abstrakcemi a malým počtem služeb pro nejzákladnější správu procesoru, vstup/výstupních zařízení, paměti a meziprocesové komunikace.
- Většina služeb nabízených monolitickými jádry (včetně např. ovladačů, významných částí správy paměti či plánování) je implementována mimo mikrojádro v tzv. serverech, jež neběží v privilegovaném režimu.
- Příklady mikrojáder: Mach, QNX (a řada dalších RTOS), L4 (pouhých 7 služeb oproti cca 600 u jader 6.x Linuxu...), ...

- Výhody mikrojáder:
  - Flexibilita – možnost více současně běžících implementací různých služeb, jejich dynamické spouštění, zastavování apod.
  - Zabezpečení – servery neběží v privilegovaném režimu, chyba v nich/útok na ně neznamená ihned selhání/ovládnutí celého OS.

- Nevýhoda mikrojáder: vyšší režie – výrazně vyšší u mikrojader 1. generace (Mach), lepší je situace u mikrojader 2. generace (např. L4: minimalismus, maximální optimalizace i s ohledem na hardware).

- Mikrojádra 3. generace: důraz na zabezpečení, virtualizaci, návrh s ohledem na možnost formální verifikace (např. seL4, ProvenCore, ...).

❖ Hybridní jádra:

- Mikrojádra rozšířená o kód, který by mohl být implementován ve formě serveru, ale je za účelem menší režie těsněji provázán s mikrojádrem a běží v jeho režimu.
- Příklady: Mac OS X (Mach kombinovaný s BSD), Windows NT (a vyšší), ...

Historie vývoje OS

❖ Je nutné znát historii, protože se opakuje... :-)

❖ První počítače:

- Knihovna podprogramů (například pro vstup a výstup) – zárodek OS.
- Dávkové zpracování: důležité je vytížení stroje, jednoduchá podpora OS pro postupné provádění jednotlivých úloh seřazených operátory do dávek (OS GM-NAA I/O, IBM 704, 1956).
- Multiprogramování – více úloh zpracovávaných současně:
  - Překrývání činnosti procesoru a vstup/výstupního podsystému (vyrovnávací paměti, přerušení).
  - Zatímco jedna úloha běží, jiná může čekat na dokončení I/O (problém: ochrana paměti, řešeno technickými prostředky).
  - OS začíná být významnou částí programového vybavení (nevýhoda: OS zatěžuje počítač).

❖ Příchod levnějších počítačů:

- Interaktivnost, produktivita práce – lidé nečekají na dokončení zpracování dávky.
- Stále se ještě nevyplatí každému uživateli dát počítač – terminály a sdílení času: timesharing/multitasking, tj. „současný“ běh více aplikací na jednom procesoru.
- Problém odezvy na vstup: preemptivní plánování úloh[a].
- Oddělené ukládání dat uživatelů: systémy souborů.
- Problémy s přetížením počítače mnoha uživateli.
- OS řídí sdílení zdrojů: omezené použití prostředků uživatelem (priority, quota).

---

[a] *Nepreemptivní plánování: Procesor může být procesu odebrán, pokud požádá jádro o nějakou službu (I/O operaci, ukončení, vzdání se procesoru). Preemptivní plánování: OS může procesu odebrat procesor i „ proti jeho vůli“ (tedy i když sám nežádá o nějakou službu jádra), a to na základě příchodu přerušení při určité události (typicky při vypršení přiděleného časového kvanta, ale také dokončení I/O operace jiného procesu apod.).*

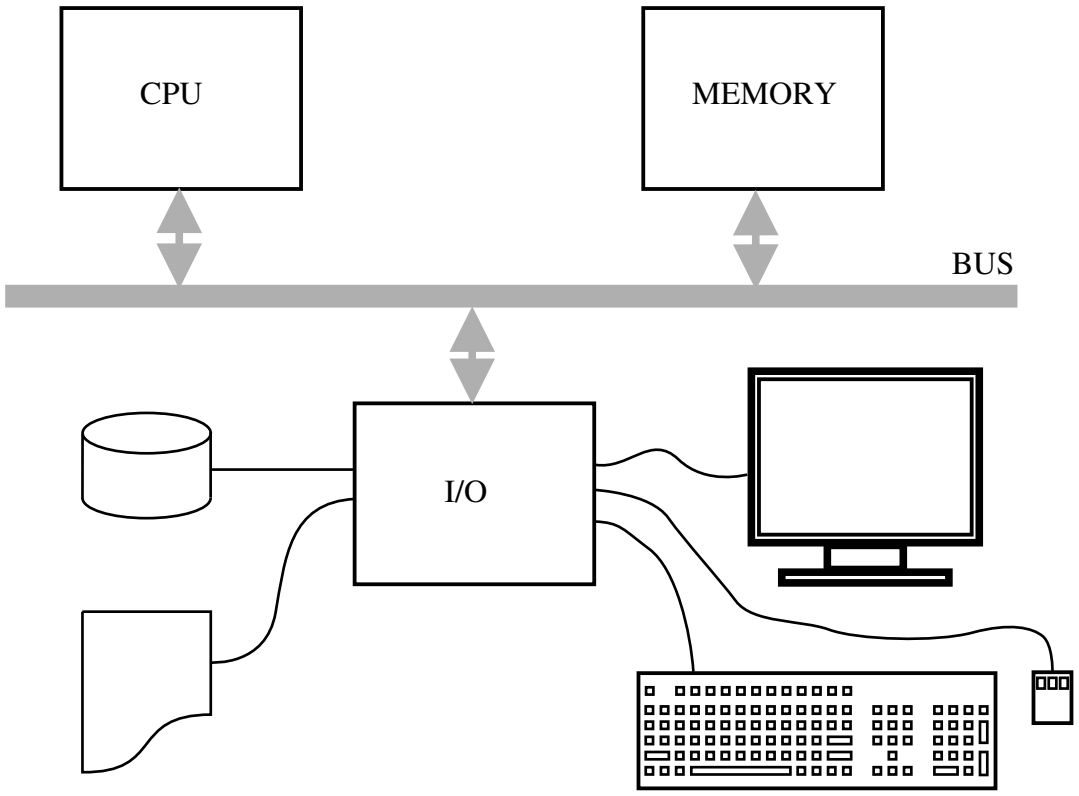
❖ Ještě levnější počítače:

- Každý uživatel má svůj počítač, který musí být levný a jednoduchý (omezená paměť, chybějící ochrana paměti, jednoduchý OS – na poli OS jde o návrat zpět: CP/M, MS-DOS).
- Další pokrok v technologiích – sítě, GUI.
- Nové OS opět získávají vlastnosti starších systémů (propojení přes síť si opět vynucuje patřičné ochrany, ...).

❖ Následoval (následuje) podobný vývoj u nejprve málo výkonných notebooků, kapesních počítačů a mobilů s omezenými OS (jednoúlohovost apod.), nyní již ale běžně s více-jádrovými procesory a převzetím mnoha rysů běžných OS.

❖ Další podobný vývoj: např. vestavěné systémy, senzorové sítě, Internet of Things (IoT) apod. – prozatím málo výkonné uzly (spotřeba energie), ale co bude následovat?

Přehled technického vybavení



Procesor: řadič, ALU, registry (IP, SP), instrukce, (navíc případně koprocesor(y))

Paměť: adresa, hierarchie pamětí (cache, ...)

Periferie: disk, klávesnice, monitor, (I/O porty, paměťově mapované I/O, přerušení, DMA)

Sběrnice: FSB, HyperTransport, QPI, UPI, NVLink, CAPI, PCI, USB, IEEE1394, ATA/SATA, SCSI/SAS, ...

Klasifikace počítačů

- Klasifikace počítačů podle účelu:
  - univerzální,
  - specializované:
    - vestavěné (řízení technologických zařízení, palubní počítače, spotřební elektronika, ...),
    - aplikačně orientované (databázové, výpočetní, síťové servery, ...),
    - vývojové (zkoušení nových technologií), ...

- Klasifikace počítačů podle výkonnosti:
  - vestavěné počítače
  - tablety, mobily, televize
  - osobní počítače (personal computer)/pracovní stanice (workstation),
  - servery,
  - střediskové počítače (mainframe),
  - superpočítače.

Klasifikace OS

- Klasifikace OS podle účelu:
  - univerzální (Windows, Linux, UNIX, ...),
  - specializované:
    - real-time – RTOS (RTEMS, FreeRTOS, PikeOS, QNX, ...),
    - vestavěné (RTOS a/nebo upravený Linux, *BSD, Windows Embedded/IoT, ...),
    - databáze, web, ... (např. z/VSE),
    - mobilní zařízení (Android, iOS, ...), ...
    - herní konzole (Orbis OS, Xbox system software, ...)

- Klasifikace OS podle počtu uživatelů:
  - jednouživatelské (CP/M, MS-DOS, ...) a
  - víceuživatelské (UNIX, Windows, ...).

- Klasifikace OS podle počtu současně běžících úloh:
  - jednoúlohové a
  - víceúlohove (multitasking: ne/preemptivní).

Příklady dnes používaných OS

typ počítače	příklady OS
mainframe	z/OS, z/VM, z/VSE, varianty Linuxu, z/TPF
superpočítače	varianty Linuxu
server	Windows, Linux, FreeBSD, UNIX
PC	Windows, MacOS X, Linux
real-time	RTEMS, FreeRTOS, PikeOS, QNX
vestavěné	různé RTOS, Linux, *BSD, Windows IoT
tablety, mobily, hodinky	Android, iOS, Tizen, Wear OS

Implementace OS

- OS se obtížně programují a ladí, protože to jsou
  - implementace různorodých a složitých funkcí,
  - (často) velké programové systémy,
  - paralelní a asynchronní systémy,
  - systémy závislé na technickém vybavení.

- Z výše uvedeného plyne:
  - Jistá setrvačnost při implementaci: snaha neměnit kód, který již spolehlivě pracuje.
  - Používání řady technik pro minimalizaci výskytu chyb, např.:
    - inspekce zdrojového kódu (důraz na srozumitelnost!),
    - rozsáhlé testování,
    - podpora vývoje technik automatizované statické/dynamické analýzy a verifikace, včetně technik formálních či s formálními základy.

Hlavní směry ve vývoji OS

- Pokročilé architektury: mikrojádra (zdůraznění výhod, minimalizace nevýhod), možnost kombinace různých, současně běžících OS, ...
- Bezpečnost a spolehlivost.
- Multiprocessing, podpora mnoha jader, podpora koprocesorů.
- Virtualizace.
- Distribuované zpracování, clustery, gridy, cloud, kontejnery, Internet of Things.
- OS tabletů, mobilů, vestavěných systémů, ...
- Vývoj nových technik návrhu, implementace a verifikace OS.
- ...

Základy práce s UNIXem

Studentské počítače s UNIXem na FIT

- studentské UNIXové servery: eva (FreeBSD), merlin (Linux/CentOS)
- PC s Linuxem (CentOS): běžně na učebnách (dual boot)

❖ Přihlášení:

```
Login: xnovak00
Password:       # neopisuje se - změna: příkaz passwd
...
$               # prompt - vyzývací znak shellu
```

❖ Vzdálené přihlášení:

- programy/protokoly pro vzdálení přihlášení: ssh, telnet (telnet neužívat!)
- putty – ssh klient pro Windows
- Cygwin: https://www.cygwin.com – kolekce *UNIX-like* nástrojů pro Windows (termínál, shell, ssh, ...)

Základní příkazy

❖ Práce s adresáři a jejich obsahem:

- `cd` – change directory
- `pwd` – print working directory
- `ls [-al]` – výpis obsahu adresáře/informace o souborech

- `mkdir` – make a directory
- `rmdir` – remove a directory

- `mv` – přesun/přejmenování souboru/adresáře (move)
- `cp [-r]` – kopie souboru/adresáře (copy)
- `rm [-ir]` – smazání souboru/adresáře (remove)

❖ Výpis obsahu souboru:

- `cat` – spojí několik souborů na std. výstup
- `more`/`less` – výpis po stránkách
- `head -13 FILE` – prvních 13 řádků
- `tail -13 FILE` – posledních 13 řádků

- `file FILE` – informace o typu obsahu souboru

Speciální znaky

- `^C` – ukončení procesu na popředí
- `^Z` – pozastavení procesu na popředí (dále `bg`/`fg`)
- `^S`/`^Q` – pozastavení/obnovení výpisu na obrazovku
- `^\` – `^C` a výpis „core dump“

- `^D` – konec vstupu (`$^D` ukončí shell)

- `^H` – smazání posledního znaku (backspace)
- `^W` – smazání posledního slova
- `^U` – smazání současného řádku