

Ako sa za 24h NEnaučiť na zápočet

Minulý rok bola databáza 40 otázok, jeden zápočet máme v rámci FX.devel jeden mám ešte ja

Zápočet FX.devel → Test 1

Grežo Cviko 4

Grežo cviko 5

Moje poznámky z cvík Grežo: 1-5

OS, ez

Otázky zo zápočtu, kt. mám ja




Ak to chce niekto testovať tak hlavne tie príkazy nedávajte do toho školského linuxu!!
história príkazov je vidno..

Test 1

Test 2

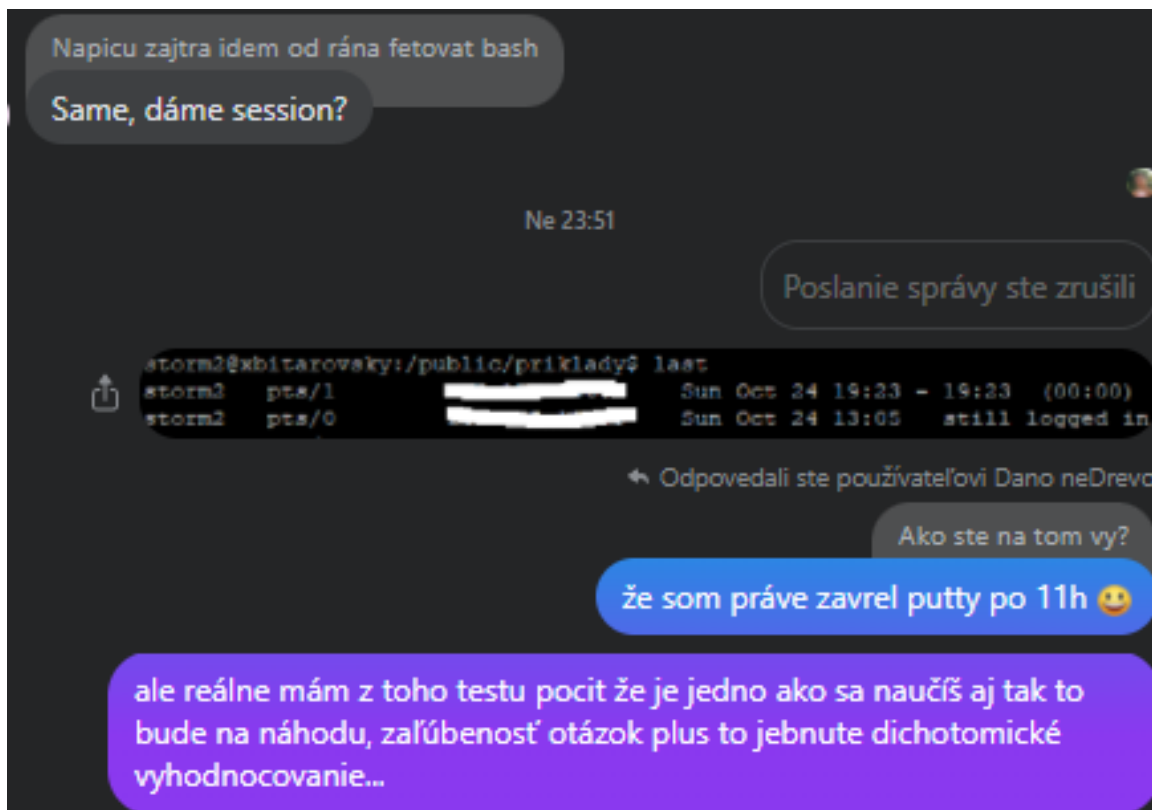
▼ Mood

**Romanko** Yesterday at 11:48 PM
@FIIT doplnil som poznámky z cvika 1-4 z OS, <https://www.notion.so/tentibor/OS-ez-e173495695eb4c50892fe32be9090490>
ešte mi chýba cviko 5 ale to až zajtra lebo teraz mám ešte inú prácu..

The Workspace on Notion

OS, ez

A new tool for teams & individuals that blends everyday work apps into one.



Good to know 1

```
storm2@xbitarovsky:/etc$ help exec
exec: exec [-cl] [-a name] [command [argument ...]] [redirection ...]
Replace the shell with the given command.
```

Execute COMMAND, replacing this shell with the specified program. ARGUMENTS become the arguments to COMMAND. If COMMAND is not specified, any redirections take effect in the current shell.

Options:

- a name pass NAME as the zeroth argument to COMMAND
- c execute COMMAND with an empty environment
- l place a dash in the zeroth argument to COMMAND

If the command cannot be executed, a non-interactive shell exits, unless the shell option ``execfail'` is set.

Exit Status:

Returns success unless COMMAND is not found or a redirection error occurs.

```
storm2@xbitarovsky:/etc$ help source
source: source filename [arguments]
Execute commands from a file in the current shell.
```

Read and execute commands from FILENAME in the current shell. The entries in \$PATH are used to find the directory containing FILENAME. If any ARGUMENTS are supplied, they become the positional parameters when FILENAME is executed.

Exit Status:

Returns the status of the last command executed in FILENAME; fails if FILENAME cannot be read.

Good to know 2

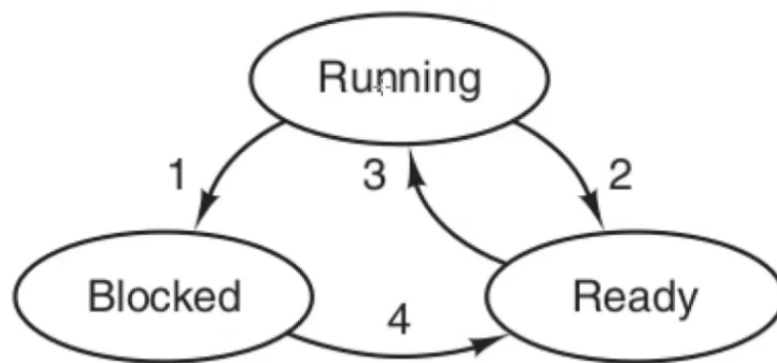
fork() a exec() - Unix

V Unixe proces vzniká systémovým volaním fork(). fork() vytvorí presnú kópiu procesu so všetkými pridelenými prostriedkami. Pamäť programu nie je nutné kopírovať a rodič s dieťaťom ju zdieľajú. Ak chceme spustiť iný program Unix používa volanie exec(). exec() nahradí aktuálny program procesu novým programom.

spawn() - Windows

Windows používa volanie CreateProcess(). Toto volanie na základe parametrov vytvorí úplne samostatný proces.

- Dobrovoľný zánik:
 - Štandardné ukončenie programu.
 - Štandardné ukončenie s chybou.
- Nedobrovoľný zánik:
 - Ukončenie s fatálnou chybou. (Segmentation fault, division by zero)
 - Ukončenie iným procesom (Kill)



- ❶ Proces je blokový lebo čaká na udalosť.
- ❷ Proces je preplánovaný OS.
- ❸ Proces je naplánovaný OS.
- ❹ Proces je zobudený po príchode udalosti.

Preemptive

plánovanie umožňuje prerušenie vykonávanej úlohy plánovačom.

Non-preemptive

plánovanie dovoľuje nahradiť vykonávaný proces len ak sa on dobrovoľne rozhodne alebo ak je blokový systémovým volaním.

Kritériá plánovania

- Spoločné
 - Férovosť - každý proces by sa mal dostať k CPU.
 - Vynucovanie politiky - politika plánovania je dodržiavaná vždy.
 - Vyváženosť - Celý systém je rovnako zaneprázdnený.
- Batch systémy
 - Throughput - maximálny počet úloh spracovaných za čas.
 - Turnaround time - doba vykonania úlohy je čo najkratšia.
 - CPU využítie - CPU je vyťažené neustále.
- Interaktívne systémy
 - Doba odozvy - Požiadavky spracované čo najrýchlejšie.
 - Proporcionalita - Splňa očakávania používateľa.
- RT systémy
 - Dodržanie doby odozvy - Zamedzenie strate dát.
 - Predvídateľnosť - Plánovač sa správa rovnako za rovnakej situácie.

User space threads

- Knižnica s funkciami, ktoré neobsahujú systémové volanie
- Kernel OS nie je volaný - rýchle operácie nad vláknami
- Tabuľka vlákien je udržiavaná v kontexte procesu.
- Kernel OS nemusí podporovať vlákna.
- Každý proces môže implementovať vlastný plánovací algoritmus.
- Neexistuje možnosť prepínať vlákna pomocou OS.
- Vláknko ktoré sa vykonáva musí používať yield().
- Vláknko blokovanie na systémovom volaní blokuje celý proces.

Kernel space threads

- Kernel implementuje podporu vláken.
- Operácie nad vláknami obsahujú systémové volanie.
- Kernel sprostredkuje plánovanie vláken.
- Kernel umožňuje prepnutie vlákna bez potreby volania `yield()`.
- Vláknko blokované na systémovom volaní neblokuje celý proces.
- Vytvorenie a ukončenie vlákna je rádovo pomalšie. (recyklácia?)

Čo vieme že bude na zápočte

find

grep

A terminal window with a dark background. The prompt is 'rgrezo@rgrezo:~/priklady\$'. The command entered is '< << <<< > |& ... 2>&1 ...'.

```
rgrezo@rgrezo:~/priklady$ < << <<< > |& ... 2>&1 ...
```

tieto kokotiny

zoznamy nejaké či čo to resp. to sliceovanie array či čo to dačo ako v pythone ale funguje to trochu inak

interpreter, aj treba vedieť čo je bc

info ohľadom ps, ako aj oživenie procesu a to posúvanie

TEST

(može byť možno)

- všetky veci v príkladoch 1.sh až 9.sh
- find, grep
- základné príkazy ls, cd, pwd, man, less, more, substitúcia historia (!1),
- who, last, jobs, .profile, .bashrc, substitúcia cez \$, jednoduchá pipe, presmerovanie vstupu < << <<< > ||& 2>&1 a podobne kokociny
- ps = výpis procesov, zastavenie procesu, ozivenie procesu
- /dev/null /dev/urandom
- budú 4 levely, 1 level teória a ďalšie 3 levely budú otázky z terminálu, bude sa len klikat, bude potreba označiť aj nesprávne a aj správne
- otázka: v otázke sú 4 body = musia všetky 4 body správne označené pre získanie bodu
- man bash si prečítať

Hľadať si v histórii o príkazoch veci (nagrepovať si riadky s grepom, riadky s manom)

jebem nato 🤔