**SJF algoritmus:**

SJF = Shortest Job First (alebo aj RR – Round Robin)

Predpokladá znalosť dĺžky požiadaviek na dávku CPU pre každý proces. Vyberá sa proces s najkratšou požiadavkou na CPU. SJF je optimálny algoritmus (dáva minimálnu priemernú dobu čakania). Môže mať Preemtívnu a Nepreemtívnu formu.

- Nepreemptívny Algoritmus - Neumožňuje predbiehanie. Akonáhle sa CPU poskytne vybranému procesu, tento proces nemôže byť predbehnutý žiadnym iným procesom, pokiaľ pridelenú dávku CPU nedokončí.

- Preemptívny Algoritmus - Umožňuje predbiehanie. Keď sa vo fronte pripravených (ready) procesov objaví proces s kratšiou požiadavkou na dĺžku dávky CPU ako je doba k dokončeniu dávky práve bežiaceho procesu, je bežiacy proces prerušený a CPU sa prideľuje novému procesu. Nazýva sa aj SRTF – Shortest Remaining Time First.

Príklady :

**Nepreemtivní algoritmus:** P1 vznik v čase 0, trvaní 7

P2 vznik v čase 2, trvaní 4

P3 vznik v čase 4, trvaní 1

P4 vznik v čase 5, trvaní 4

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| T1 | T2 | T3 | T4 | T5 | T6 | T7 | T8 | T9 | T10 |
| P1 | P1 | P1 | P1 | P1 | P1 | P1 | P3 | P2 | P2 |
|  | P2 – T4 |  | P3 – T1 | P4 – T4 |  |  |  |  |  |

Ktorý proces pobeží v čase 10? ­> P2

Nepreemptivní SJF - bez preskakovania

P1,P3,P2,P4 -> 0-7,7-8,8-12,12-16

**Preemptivní algoritmus:** P1 vznik v čase 0, trvaní 7

P2 vznik v čase 2, trvaní 4

P3 vznik v čase 4, trvaní 1

P4 vznik v čase 5, trvaní 4

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| T1 | T2 | T3 | T4 | T5 | T6 | T7 | T8 | T9 | T10 |
| P1 | P1 | P2 | P2 | P3 | P2 | P2 | P4 | P4 | P4 |
|  |  | P2 – T4 | P3 – T1 | P5 – T4 |  |  |  |  |  |

Ktorý proces pobeží v čase 10? > P4

Preemptivní SJF - s preskakovaním

P1,P2,P3,P2,P4,P1 -> 0-2,2-4,4-5,5-7,7-11,11-16

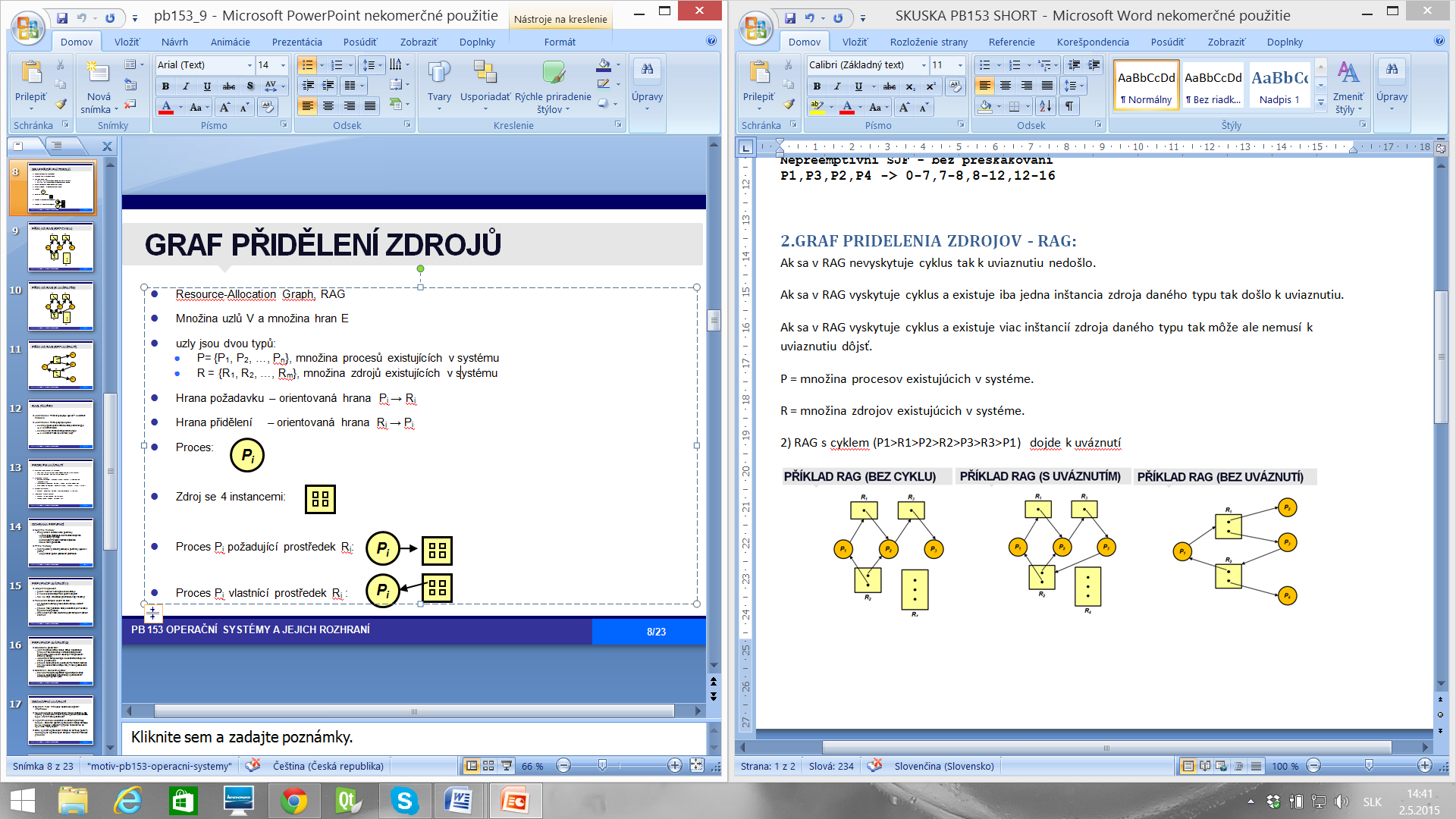
**GRAF PRIDELENIA ZDROJOV - RAG:**

Ak sa v RAG nevyskytuje cyklus tak k uviaznutiu nedošlo.

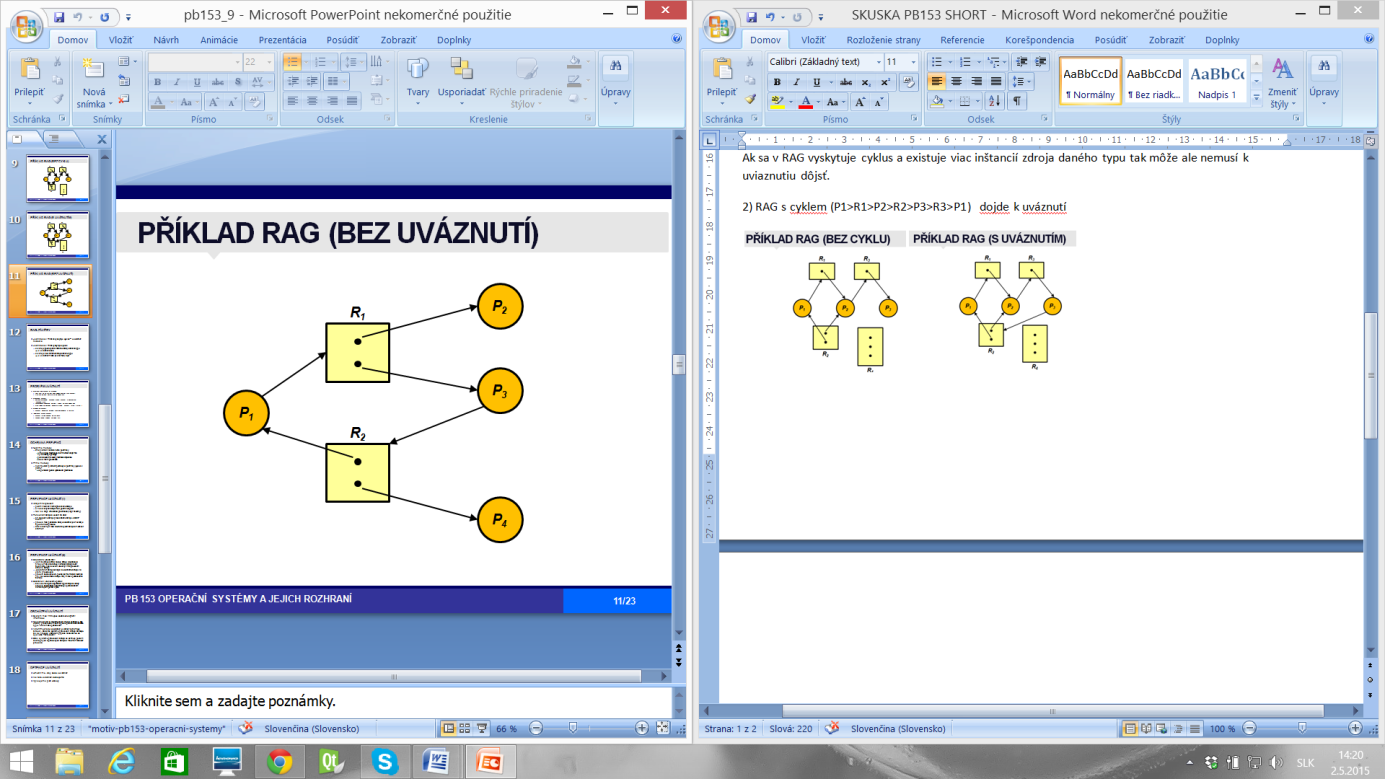
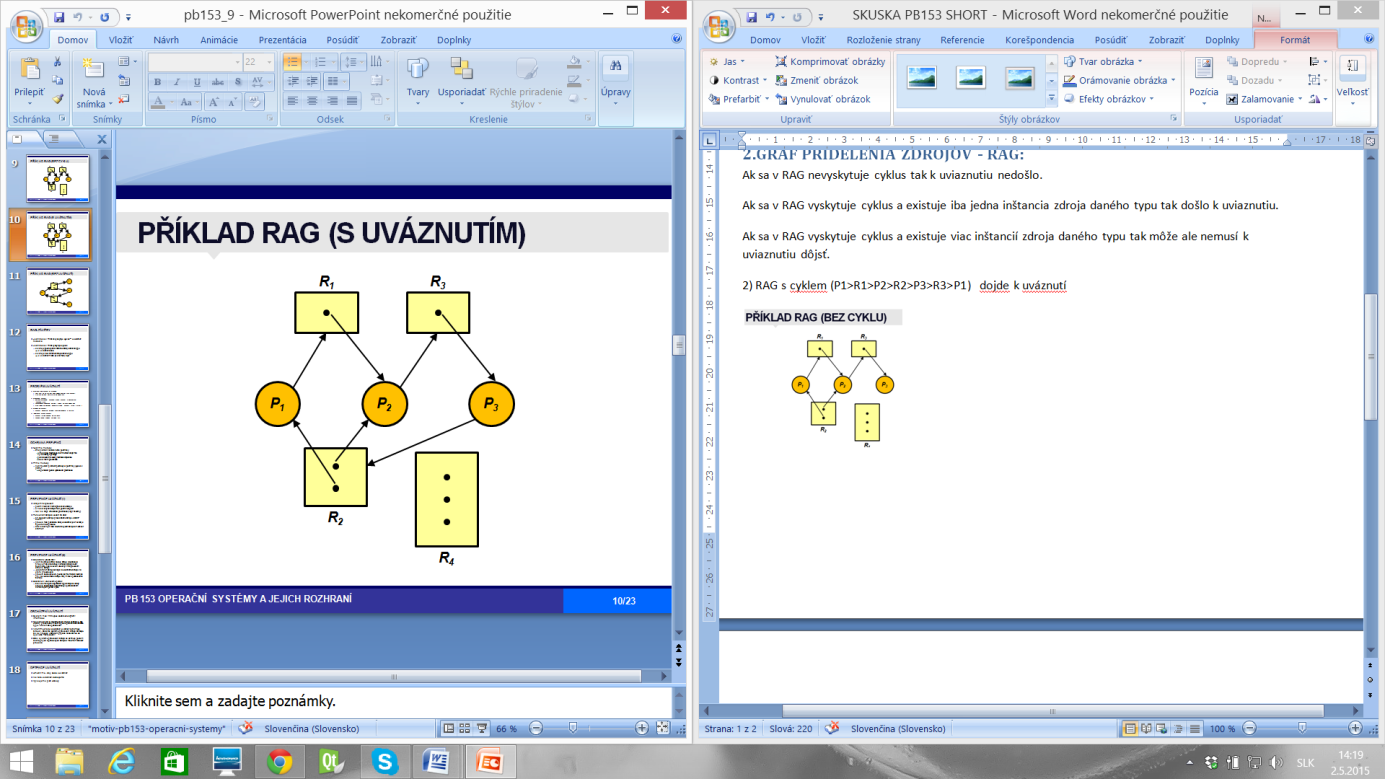
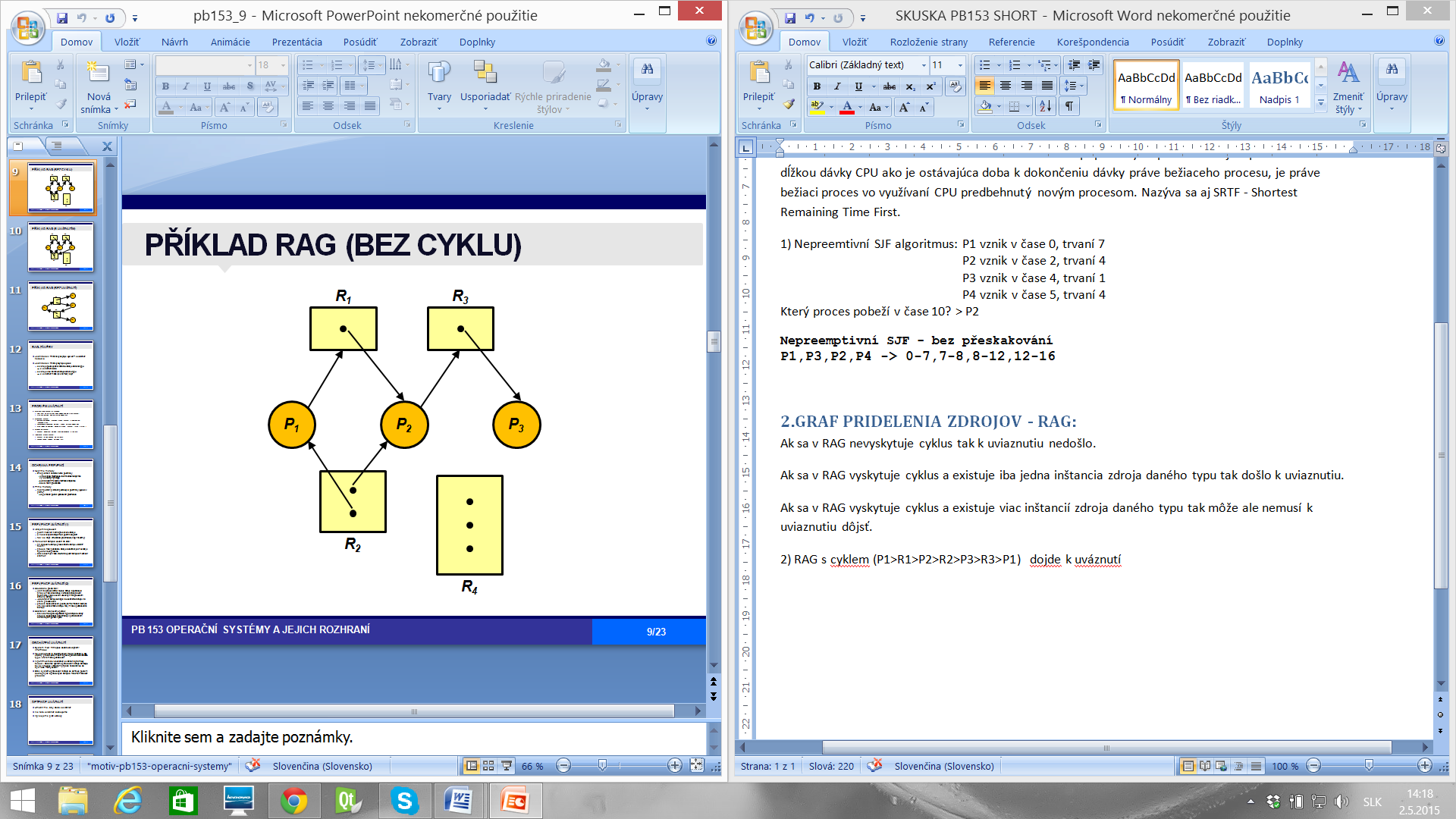
Ak sa v RAG vyskytuje cyklus a existuje iba jedna inštancia zdroja daného typu tak došlo k uviaznutiu.

Ak sa v RAG vyskytuje cyklus a existuje viac inštancií zdroja daného typu tak môže ale nemusí k uviaznutiu dôjsť.

P = množina procesov existujúcich v systéme. R = množina zdrojov existujúcich v systéme.



RAG s cyklem (P1­>R1­>P2­>R2­>P3­>R3­>P1) ­ dôjde k uviaznutiu



P1->R1->P2->R3->P3 ==> bez cyklu bez uviaznutia

P1->R1->P2->R3->P3->R2.1->P1 && R2.2->P2 ==> s cyklom s uviaznutím

P1->R1.1->P2&&R1.2->P3->R2.1->P1&&R2.2->P4 ==> s cyklom bez uviaznutia

P1->R1->P2->R2->P3->R3->P1 ==> s cyklom s uviaznutím

**IPC Medziprocesová Komunikácia, Príklady :**

IPC – Inter Process Communication, alebo medziprocesová komunikácia – Procesy medzi sebou komunikujú signálmi, napr. asynchrónnymi udalosťami, pomocou rúr, správami, semaformi, či skrz zdieľanú pamäť.

**Mikrojadro, Definuj, Výhody+ Nevýhody :**

Malé jadro OS plniace iba niekoľko základných, potrebných funkcií . Zodpovedá napr. za primitívnu správu pamäte, či komunikáciu medzi procesmi. Väčšia časť funkcionality jadra sa presúva do užívateľskej oblasti - ovládače HW zariadení, služby súborového systému, virtualizácia pamäte... Medzi užívateľskými procesmi sa komunikuje posielaním správ.

Výhody:

-Ľahká prenositeľnosť OS. Jadro je malé, stabilnejšie.

-Vyššia spoľahlivosť . Moduly majú jasné API a sú ľahko testovateľné.

-Vyššia bezpečnosť . Menej kódu OS beží v režime jadra.

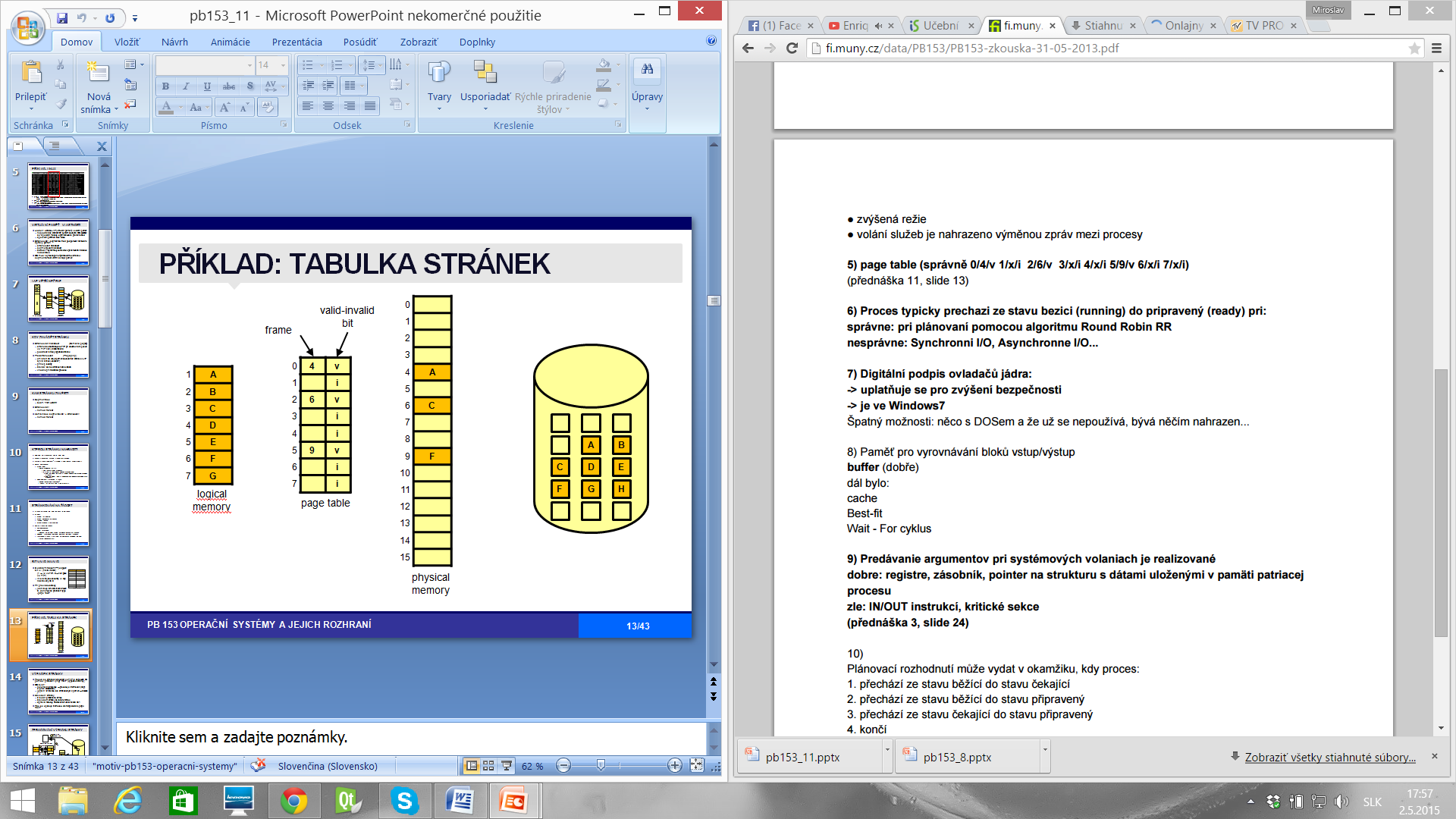
-Vyššia Flexibilita . Umožňuje jednoduchšiu modifikáciu, pridanie či odoberanie modulov.

-Všetky služby sú poskytované jednotne , výmenou správ.

Nevýhoda:

Zvýšená réžia – Jednoduché volania služieb sú nahradené výmenou správ medzi procesmi.

**Stránkovanie pamäte, Opíš + Graf :**

Stránkovanie je adresovacia technika slúžiaca na zmysluplnú adresáciu pamäte. Fyzická pamäť sa podľa hardwarových možností rozdelí na úseky pevnej dĺžky, tzv. Rámce. Logický adresový priestor sa rozdelí na rovnako veľké úseky, tzv. Stránky. Proces stránkovania je prideľovania rámcov pamäte jednotlivým stránkam. Vizualizácia sa realizuje pomocou tzv. Page Table :

**Stavy procesov a prechody medzi nimi, Opíš + Graf :**

Proces sa môže nachádzať celkovo v piatich stavoch:

-New – Nový, práve vytvorený proces.  
-Ready – Proces čaká na pridelenie časovej dávky procesoru

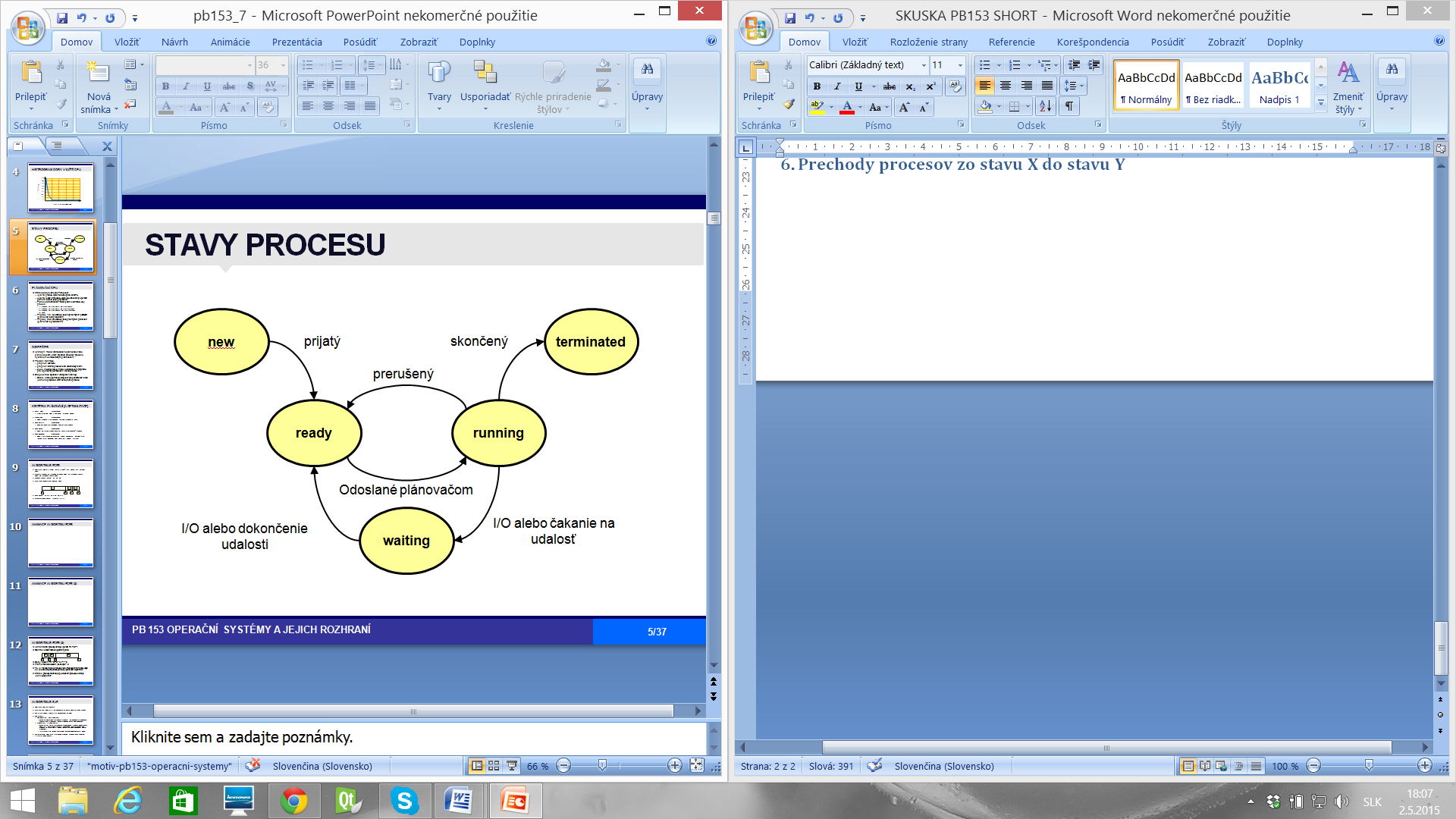
-Waiting – Proces čaká na udalosť

-Running- Proces je práve vykonávaný niektorým procesorom

-Terminated- Ukončený proces

Pre synchrónne I/O – Proces typicky prechádza zo stavu bežiaci (running ) do stavu čakajúci (waiting).

Pri plánovaní pomocou algoritmu RR – Proces prechádza zo stavu bežiaci (running) do stavu pripravený (ready), keď ho algoritmus dočasne odstaví od procesoru.



**Proces, Správa Procesov. Opíš :**  
Proces je spustený program. Jeden proces môže byť zároveň spustený viacerými uživateľmi. Je taktiež jednotkou práce systému, pre svoju činnosť potrebuje zdroje.

Typické aktivity OS v oblasti správy procesov – Vytváranie a Ukončovanie procesov, Potláčanie a Obnovovanie procesov, Mechanizmy pre komunikáciu a synchronizáciu medzi procesmi , Mechanizmy na detekciu a riešenie uviaznutia (deadlock).

**Plánovacie rozhodnutie , kedy sa vydáva. Situácie :**

**-** Nepreemtívne plánovanie (Plánovanie bez predbiehania):

1. Proces prechádza zo stavu bežiaci (running) do stavu čakajúci (waiting)

2. Proces prechádza do stavu skončený (terminated)

-Preemptívne Plánovanie (Plánovanie s predbiehaním):

1. Proces prechádza zo stavu bežiaci (running) do stavu pripravený(ready)

2. Proces prechádza zo stavu čakajúci (waiting) do stavu pripravený (ready)

**Plánovacie algoritmy na Linux-e, Názvy :**  
Deadline

Nope  
Anticipatory  
CFQ

**Digitálny podpis ovládačov jadra, Opíš :**

Podpisovanie u Windows – garantuje spoľahlivosť ovládačov, systému a tým pádom aj bezpečnosť jadra. Nejde o čistotu jadra ani o licencie. Nachádza sa napr. vo Win7.

Podpisovanie v Linux-e – Dostupné od roku 2004, ale nepoužíva sa príliš často. Linux moduly zvyčajne nepodpisuje, ale všíma si licencie.

Pamäť pre vyrovnávanie blokov vstupu/výstupu **. Názov + Opíš :**

Vyrovnávacia pamäť sa nazýva buffer.  
Používa sa väčšinou ako spoj medzi prvkami systému s rozdielnou rýchlosťou spracovávania informácií. V prípade I/O zariadení slúži ako dočasné ´´úložisko´´ pre dáta, ktoré idú von, resp. Dnu, avšak ešte stále čakajú na spracovanie kvôli hardwarovým rýchlostným obmedzeniam.

**Vlákna. Definuj, Výhody:**

Vlákno je objekt vznikajúci v rámci procesu, viditeľný iba vo vnútri procesu a charakterizovaný svojim stavom. Model Procesy a Vlákna – Proces je jednotkou vlastniacou zdroje, Vlákno je jednotkou plánovania činností.

Každé vlákno si udržuje svoj vlastný : zásobník, program counter, registre a TCB (thread context block).

Vlákna zdieľajú: pamäť (adresový priestor), číslo procesu, číslo vlákna.   
Vlákno môže pristupovať k pamäti a zvyšným zdrojom svojho procesu. Zdroje procesu zdieľajú všetky vlákna daného procesu. Súbor otvorený jedným vláknom majú k dispozícii všetky ostatné vlákna.

Výhody Vlákien:

- využitie multiprocesorových strojov (vlákna jednoho procesu môžu zároveň bežať na rôznych CPU), napr: jedno vlákno prevádza používateľom definovanú úlohu a druhé vlákno prekresľuje obrazovku.

- vlákno sa vytvorí rýchlejšie ako proces

- vlákno sa ukončí rýchlejšie ako proces

- medzi vláknami sa prepína rýchlejšie, ako medzi procesmi

- jednoduchšia štruktúra programu

Vytváranie Vlákien:  
UNIX – pthread\_create()  
Windows – CreateThread()

**Fork(), Definuj :**

LINUX-ová funkcia vytvorí nový proces. Proces je kópiou rodiča, Volanie tejto funkcie je implementované ako Copy-On-Write, t.j. kým nemeníme pamäť, tak sa iba zdieľa a skutočná kópia sa vytvorí až pri pokuse o modifikáciu.

c**reateProcess(), Definuj :**

Využívaná vo Win API, ekvivalent LINUX-ovej funkcie Fork()

**Predávanie argumentov pri systémových volaniach , Spôsoby :**

Pri systémových volaniach sa predávanie argumentov realizuje registrami, zásobníkom, blokom v pamäti či pointerom na štruktúru s dátami uloženými v pamäti patričného procesu.

**Systémové volania, Opíš s ohľadom na UNIX, Windows :**

Tvoria rozhranie medzi užívateľským rozhraním a OS. Je možné ich implementovať pomocou prerušenia.Typicky popísané ako inštrukcie v assembleri a uvádzané v užívateľskom manuáli OS. Existujú štandardy uľahčujúce prenositeľnosť programového kódu - v prípadeUNIXu: POSIX, v prípade Windows: Win32 (Win API).

Príkladom systémového volania sú napr. LINUXové volania open, close, exit

Brány systémových volaní:

- v MS-DOS: int 0x21

- v LINUX štandardne 0x80, novšie aj (sysenter/syscall)

Sysenter -Slúži k systémovému volaniu, optimalizované pre rýchlosť

**POSIX , Definuj:**Portable Operating System Interface. Je to prenosné rozhranie pre operačné systémy, štandardizované ako IEEE 1003 a ISO/IEC 9945. Vychádza zo systémov UNIX a určuje, ako majú POSIX-kompatibilné systémy vyzerať, čo majú vedieť, obsahovať, akým spôsobom majú vykonávať určité úlohy a podobne.

**FREEBSD, Opíš :**  
Systémové volanie realizované cez volanie funkcie, alebo cez volanie brány call 7:0, ktorá obsahuje int 0x80. Číslo aj výsledok systémového volania je v registri eax. Viaže sa na zásobník, na ktorom sa nachádzajú parametre systémového volania ( 1 parameter uložený ako posledný). Volajúci proces musí po svojom skončení zásobník vyčistiť.

**MS-DOS v kontexte spúšťania procesov**  
Neumožňuje spúšťať procesy ´´paralelne´´, teda sa jedná o Single- Tasking Systém. Využíva opisovač procesu (Program Segment Prefix)

**RES, Definuj :**

Fyzická pamäť rezidentná v čase, ktorá po použití procesom nebola odswapovaná.

**Real time systémy , Opíš :**

Systémy optimalizované pre výkon funkcie v reálnom čase, väčšinou zamerané na špecifický set akcií – napr. riadenie strojov, výrobnej linky...

**Paralelný vs distribuovaný systém :**

Paralelné systémy sú úzko previazané. Skladajú sa z niekoľko vzájomne komunikujúcich CPU, zdieľajúcich jednu pamať a synchronizujúcich sa podľa spoločného hodinového signálu. Výhodou je vyššia priepustnosť systému a ekonomické využitie PC. Delia sa na dve podtriedy:

**-**SMP **–** Symetrický Multiprocesorový Systém. Procesory sú si rovné, na každom beží rovnaká kópia OS. V súčasnosti podporované Mac OS, Linux, Windows, FreeBSD.

**-** AMP **-** Asymetrický Multiprocesorový Systém. Procesory si nie sú rovné, každý rieši špecifický set úloh – Napr. jeden procesor plánuje prácu ostatným, alebo určité typy procesov sú vyhradené len špecifickým procesorom.

Distribuované systémy – Každá CPU má vlastnú pamäť, pracuje sám za seba. Komunikácia prebieha pomocou komunikačných spojov. Výhodou je zdieľanie zdrojov a vyššia spoľahlivosť, nevýhodou neefektívnejšie využitie priepustnosti. Využívajú architektúry Peer-To-Peer, Server- Klient.

**Multitasking /Timesharing, Definuj :**

Multiplikovanie CPU. V skutočnosti sa naraz vykonáva len jedna úloha, ale CPU sa medzi úlohami podľa potreby prepína, takže užívateľ získava dojem, že sú spracovávané paralelne. Multitasking umožňuje uživateľom zdieľať počítačový systém.

Prekladaním behu procesov maximalizujeme využitie procesoru a minimalizujeme dobu odpovede. (response time). Oproti multiprogramovaniu znižuje dobu odozvy (response time) interaktívnych procesov.

Multitaskingové systémy sú značne komplexné.

Multiprogramovanie –Optimalizované pre efektivitu využitia CPU, ktorú značne zvyšuje. Má však relatívne vysokú response time (dobu odozvy) u interaktívnych aplikácií.

**Uviaznutie a jeho podmienky, Opíš :**Obsadenie je spôsobené čakaním na pridelenie zdroja, ktorý je ale obsadený.

Aby došlo k uviaznutiu, platia súčasne nasledujúce štyri podmienky:

- vzájomné vylúčenie (mutual exclusion) - zdieľaný zdroj môže v jednom okamihu používať len jeden proces

**-** ponechanie si zdroja a čakanie na ďalší (hold and wait) – Proces čaká na získanie ďalšieho zdroja, doteraz vlastneného inými procesmi, pričom stále neuvoľnil iný zdroj

- bez predbiehania (no preemption) – Nemožno dynamicky prideľovať zdroj, uvoľňuje sa až procesom, ktorý ho momentálne vlastní až potom, čo ho prestane daný proces potrebovať.

- kruhové čakanie (circular wait)

**Strategický plánovač** **– Dlhodobý** :

Vyberá proces ktorý je možné zaradiť medzi pripravené (ready) procesy. Určuje stupeň multiprogramovania. Nie je spúšťaný veľmi často, preto nemusí byť rýchly. Pri ukončení jedného procesu rozhodne, ktorú úlohu ďalej vybrať k zavedeniu do pamäte a spusteniu.

**Krátkodobý plánovač :**

Prideľuje procesor pripraveným (ready) procesom, má na starosti prechod procesu zo stavu ready do stavu running - je spúšťaný často a musí byť rýchly.

**Strednodobý plánovač** **– Taktický** :  
Vyberá proces ktorý môže byť zaradený medzi odložené procesy čakajúce na udalosť (waiting) a naopak, vyberá proces ktorý môže byť zaradený medzi pripravené procesy (ready).

**Strace, Definuj + Použitie :**Strace je diagnostický a debugovací nástroj LINUX-u. Monitoruje interakcie medzi procesmi a LINUX-ovým kernelom – to zahŕňa systémové volania, donášky signálu a zmeny v stavoch procesov. Strace je umožnená schopnosťou kernelu známou ako Ptrace s pomocou špeciálneho syscallu. Niektoré UNIX-ové systémy poskytujú iné diagnostické nástroje podobné Strace, napríklad Truss.

Ptrace - slúži pre trasovanie bežiaceho kódu programu, Ptrace je rozsiahly a veľmi robustný nástroj, dokáže sa napojiť na proces, krokovať ho, zasahovať do jeho pamäti, odchytávať signály a pod. Využívané v GNU, debugerom gdb.

**pthread\_create(), Definuj :**

Slúži na vytváranie vlákien v UNIX-ových systémoch. POSIX Threads, obvykle označováný ako Pthreads, je POSIXový štandard pre vlákno (thread). Standard, POSIX.1c, Threads rozšírenie (IEEE Std 1003.1c-1995), definuje API pre vytváranie a manipuláciu s vláknami.

**CreateThread()**, **Definuj : –**

Windowsový Ekvivalent POSIX-ovému pthread\_create(), funkcia slúžiaca na vytvorenie vlákana, podľa štandardu Win32.

Implementácia API je možná na mnohých Unix operačných systémoch podporujúcich POSIX ako napr. FreeBSD, NetBSD, OpenBSD, GNU/Linux, Mac OS X, Solaris, DR-DOS a implementácia pro Microsoft Windows taktiež existuje za použitia podsystému SFU/SUA, ktorý poskytuje nativnu implementáciu niekoľkými POSIX API, poprípade taktiež pri využití balíčku tretej strany pthreads-w32, ktorý implementuje pthreads na vrcholu stávajících Windows API.

**Typ jadra v UNIX, Uveď :**

Modulárne monolitické Jadro

**Plánovanie fibers vo Windows** :

Nie sú plánované systémom ani OS, každá aplikácia (proces) si musí vlákna určiť a spravovať osobitne s ohľadom na momentálnu situáciu procesorov.