10. Operacijski sustavi

10.1. Uloga operacijskih sustava

- Što je to operacijski sustav (OS)? Čemu služi?
  - „skup programa” / „međusloj između programa i sklopovlja”
  - olakšava korištenje računala
    - sve operacije dostupne kroz jednostavno sučelje OS-a
    - složenost sklopovlja i nekih operacija su skrivene od korisnika
      - OS ima upravljačke programe za rad sa sklopovljem
      - OS ima podsustave za složenije operacije (mreža, datoteke, ...)

10.1. Uloga operacijskih sustava (2)

- obzirom na tu ulogu, OS je vrlo bitan za sustav, možda najkritičniji dio
- do sada smo pretpostavili da OS radi sve „savršeno”
- ali i OS može unijeti dodatne odgode zbog svojih poslova
- odgode mogu biti predvidivog trajanja ili ne, ovisno o složenosti operacija koje on obavlja (npr. radi li u O(1) ili pak O(n) ili slično)
- različiti sustavi imaju različite OS-ove koji nastoje ispuniti zahtjeve prema takvim sustavima – nisu svi OS-evi za sve, nema „najboljeg OS-a”
- u upravljačkim sustavima, pogotovo onim vrlo jednostavnim koji koriste mikrokontrolere OS je vrlo jednostavan (više „biblioteka” nego OS)
  - upravljanje prekidima, napravama, vremenom
  - „integriran” s upravljačkim programom (povezuju se prije učitavanja)

Osnovni dijelovi operacijskog sustava (podsustavi)

1. upravljanje vanjskim jedinicama
2. upravljanje spremničkim prostorom
3. upravljanje vremenom
4. upravljanje dretvama/procesima
5. mehanizmi komunikacije i sinkronizacije
6. datotečni podsustav
7. mrežni podsustav

- Svojstva nekih od podsustava (3., 4., 5. i 7.) su već predstavljena u okviru problematike raspoređivanja, sinkronizacije, komunikacije i upravljanja vremenom (4. – 7. poglavlje)

10.1.1. Upravljanje vanjskim jedinicama

- Upravljanje vanjskim jedinicama obavlja se sklopovima kojima su vanjske jedinice spojene – upravljački sklop za napravu (kontroler).
  - Npr. USB naprava spojena je na računalo USB priključkom a njime upravlja USB upravljač – programi i OS komuniciraju s tim upravljačem da bi primili i poslali podatke prema napravi.
  - U nastavku se pod “vanjskom jedinicom” ili “napravom” zapravo podrazumijeva takav upravljač.
- upravljački program naprava (engl. *device driver*)
  - zna kako s napravom (njenim kontrolerom)
- upravljanje napravama:
  - radno čekanje, prekidi, izravan pristup spremniku

Upravljanje napravama „radnim čekanjem”

- OS sve radi sam
  1. ispituje status naprave (preko statusnog registra kontrolera)
  2. ako naprava ima neki novi podatak onda OS pročita taj podatak
  3. ako nema, onda OS ide dalje raditi nešto drugo ili čeka da ova naprava da podatak, ali to radi „radnim čekanjem” u petlji, ponavlja ovu operaciju od 1.
- iako se zove „radno čekanje” ne mora zaista i biti radno čekanje
  - naprave OS može periodički provjeravati ili na zahtjev
- mehanizam „radnog čekanja” se može koristiti za vrlo jednostavne naprave (kontrolere)
- nedostatak je radno čekanje tj. neefikasno korištenje procesora

Prekidi

- kad se nešto dogodi „izvana” naprava to javlja mehanizmom zahtjeva za prekid, tj. prekidom
- SRSV-ima je prekid jako bitan jer oni moraju većinom jako brzo reagirati na događaje
- obrada prekida:
  - 1. zabrani prekidanje; 2. prebaci se u prekidni način rada;
  - 3. spremi PC+RS na stog; 4. u PC stavi adresu prekidna potprograma
- u SRSV-ima bi obrade prekid trebali biti vrlo brze
- kada to ne bi bilo tako onda bi bio problem jer nisu svi događaji jednako bitni – kad se dogodi preklapanje događaja htjeli bi da bitniji budu prije obrađeni od manje bitnih, iako se možda dogode za vrijeme obrade tih drugih

Prekidi (2)

- Uobičajeni pristupi pri oblikovanju prekidnog podsustava su:
  1. obrađivati prekide redom prispijeća
  2. obrađivati prekide prema prioritetu te
  3. obrade prekida podijeliti na dva dijela: hitni dio i manje hitni dio
- Sva tri načina imaju svoju primjenu (i u kombinacijama)
- opcija 1. je dovoljna kad su obrade kratke – tada se jako malo čeka (ako)
- opcija 2. je „bolja” ali zahtijeva sklopovsku potporu
- opcija 3. je neophodna kad obrada traži puno vremena – tada i nije toliko hitna i treba omogućiti prihvat i obradu ostalih događaja
  - hitni dio: gornja polovica (top half / interrupt service routine)
  - manje hitni dio: donja polovica (bottom half / interrupt service thread)

Prekidi (3)

- što odabrati i kako oblikovati prekidni potprogram?
  - ovisno o potrebama sustava
- ograničenja na korištenje funkcija u prekidnom potprogramu
  - obzirom da se izvodi u obradi prekida taj kod NE SMIJE moći BLOKIRATI
  - svi pozivi koji mogu blokirati se ne smiju pozivati
    - ali zato često postoje neblokirajuće inačice
      - *PokušajČekati* ili *PokušajPročitati* ili slično
  - pozivi *Postavi/Povećaj* i sl. se mogu pozivati
  - ponekad nije očito da neki pozivi mogu blokirati
    - npr. write, send, malloc
  - ako treba sinkronizacija onda koristiti radno zaključavanje (spin lock)

10.1.2. Upravljanje spremničkim prostorom

- U OS-u prikazani načini:
  1. statičko
  2. dinamičko
  3. straničenje
- Za SRSV problem može biti u korištenju pomoćnog spremnika
  - disk je SPOR (čak i SSD) prema memoriji
  - ako treba nešto dohvatiti s diska, npr. u slučaju promašaja kod straničenja, program će stati dok se to ne obavi
  - ta dodatna odgoda izvođenja programa može biti kobna za upravljanje!
- Straničenje se može koristiti ALI VRLO OPREZNO
  - bez pomoćnog spremnika
  - zaključati kritične procese u memoriju (mlock, mlockall)

10.2. Operacijski sustavi posebne namjene

- ❑ što znači „posebne”? kako sustave klasificiramo?

1. osobno računalo, tablet, pametni telefon – nekritični sustavi
2. upravljačko računalo – kritični sustavi
3. ugrađeno računalo?
   - ➢ može biti kritično (npr. u automobilu, liftu)
   - ➢ može biti nekritično (npr. u biciklu, TV-u, satu)

- ❑ OS mora biti prilagođen sustavu u kojem se koristi
- ❑ U nastavku će se razmatrati:
  - ➢ OS za SRSV (RTOS; FreeRTOS, QNX, VXworks)
  - ➢ OS za nekritične sustave (Windows, Linux, Android, iOS)
  - ➢ prilagođeni OS-evi (RTLinux, Real-time Linux, Windows 10 IoT Core)

10.2.1. Operacijski sustav FreeRTOS

- Free – besplatno dostupan izvorni kod koji se može koristiti
- RTOS – OS za SRSV
- Svojstva:
  - vrlo malo zahtjeva od sklopovlja (memorija, mikrokontroler)
  - vrlo podesiv (što od njega uključiti ili ne)
  - zahtjeva dobro poznavanje načina korištenja (svojstva pojedinih sučelja!)
- OS i aplikacija se zajedno prevode u „sliku sustava”, koja se onda učita u ugradbeni sustav
  - uobičajeno za ugradbene sustave!

FreeRTOS: Isključivanje nepotrebnih dijelova jezgre

- Ušteda memorijskog prostora na svakom koraku!
- Nepotrebne jezgrine funkcije ne treba uključivati

u konfiguracijskoj datoteci (FreeRTOSConfig.h):

```
#define INCLUDE_Neka_Jezgrina_Funkcija 0
```

u kodu jezgre:

```
#if (INCLUDE_Neka_Jezgrina_Funkcija == 1)
   tip Neka_Jezgrina_Funkcija (parametri)
   {
   ...
   }
#endif /* INCLUDE_Neka_Jezgrina_Funkcija */
```

FreeRTOS: Imenovanje varijabli

- prefiksi označavaju tip varijabli i funkcija
- `x` - cijeli broj (`int`), `s` - `short`, `c` - `char`, ...
- `ux` - cijeli broj bez predznaka (`unsigned int`)
- `pux` - kazaljka na cijeli broj bez predznaka (`unsigned int *`)
- itd.

```
BaseType_t xTaskCreate (
     TaskFunction_t pvTaskCode,
     const char * const pcName,
     uint16_t usStackDepth,
     void *pvParameters,
     UBaseType_t uxPriority,
     TaskHandle_t *pvCreatedTask
);
```

FreeRTOS: Zaštita pri uporabi dijeljenih sredstava sustava

- radi minimizacije nepotrebnih operacija mnoge zaštite nisu automatski uključene u obične funkcije – potrebno ih je ugraditi u program!
- npr. korištenje sučelja malloc nije sigurno u višedretvenom okruženju, potrebno je u kod programa dodati zaštitu

```
vTaskSuspendAll();
pvReturn = malloc(xWantedSize);
xTaskResumeAll();
```

- još mogućnosti (uz pretpostavku jednoprocesorskog sustava):

`taskENTER_CRITICAL()` – zabrana dretvi i prekida manjeg prioriteta od dretve

`taskDISABLE_INTERRUPTS()`

FreeRTOS: Pokretanje raspoređivača dretvi - primjer

```
void vPocetnaFunkcija(void) {
   TaskHandle_t xHandle = NULL;

   /* Inicijalizacija, stvaranje ostalih potrebnih objekata */
   /* ... */

   /* Stvaranje bar jedne dretve (prije pokretanja raspoređivača) */
   xTaskCreate(vDretva, "IME_DRETVE", VEL_STOGA, NULL, prioritet, &xHandle);

   /* ... stvaranje ostalih dretvi i potrebnih objekata */

   /* Pokretanje prioritetnog raspoređivača */
   vTaskStartScheduler();

   /* Kontrola neće doći ovdje dok se ne zastavi raspoređivač
    * od strane neke dretve s: vTaskEndScheduler();
    * tj. najčešće se ovdje nikad ne vraća! */

   /* primjer micanja dretve iz sustava */
   if (xHandle != NULL)
      vTaskDelete(xHandle);
}
```

```
void vDretva(void *pvParameters) {
   for (;;) {
      /* Posao dretve */
   }
}
```

FreeRTOS: Manje zahtjevne dretve – niti

- u kontekstu ugradbenih sustava, dretve traže „puno” resursa: opisnik, stog
- manje zahtjevne dretve – **niti** (izvorno ***co-rutine***)
  - ako dretva odrađuje nešto periodički
  - ne treba pamtiti kontekst između tog posla
  - sve je u nekim drugim strukturama podataka, ne na stogu
- sve niti koriste isti stog!
- ali onda se ne mogu koristiti uobičajene metode sinkronizacije, komunikacije i sl.

FreeRTOS: Međudretvena sinkronizacija i komunikacija

- redovi poruka
- semafori
  - binarni
  - opći
  - binarni s nasljeđivanjem prioriteta (mutex semaphore)
- alarmi
- skupovi redova (čekati na više stvari, semafor, poruku)

FreeRTOS: Pozivi jezgrinih funkcija iz prekidnih funkcija

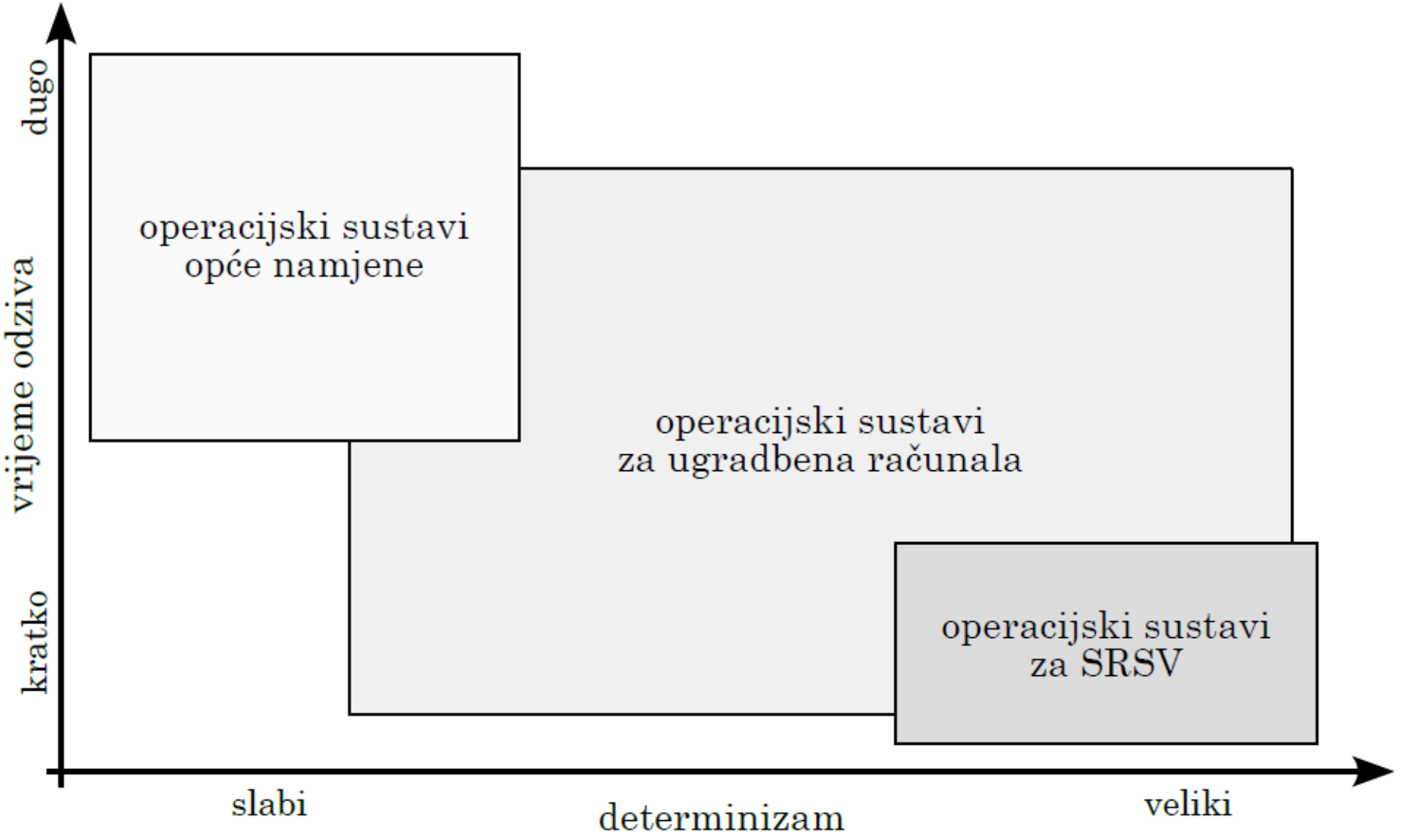
- postoje posebno prilagođene jezgrine funkcije za poziv iz obrade prekida
- sufix `FromISR`, npr. `xQueueSend` => `xQueueSendFromISR`

Usporedba funkcija:

```
BaseType_t xQueueSend (
   QueueHandle_t xQueue,
   const void * pvItemToQueue,
   TickType_t xTicksToWait
);

BaseType_t xQueueSendFromISR (
   QueueHandle_t xQueue,
   const void *pvItemToQueue,
   BaseType_t *pxHigherPriorityTaskWoken
);
```

```
void vPrekidnaFunkcija(void) {
        QueueHandle_t xQueue = dohvati_red();
        const void *pvItemToQueue = dohvati_poruku();
        BaseType_t pxHigherPriorityTaskWoken;

        xQueueSendFromISR(xQueue, &pvItemToQueue,
                &pxHigherPriorityTaskWoken);

        if (pxHigherPriorityTaskWoken) {
                portSAVE_CONTEXT();
                vTaskSwitchContext();
                portRESTORE_CONTEXT();
        }
        asm volatile ( "reti" );
}
```

10.3. Svojstva različitih tipova operacijskih sustava



Slika 10.2. Vremenska svojstva različitih sustava

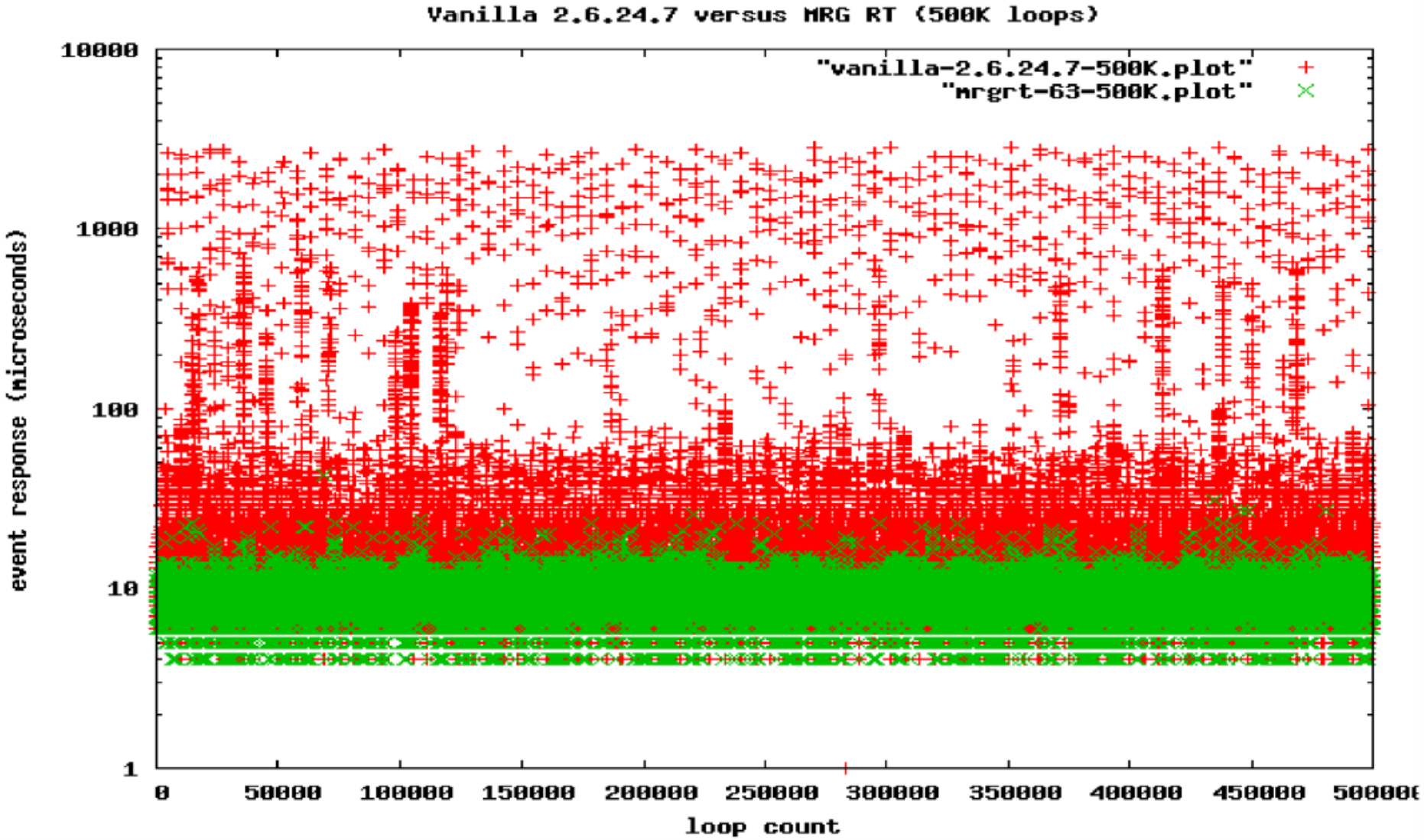
10.3.1. Operacijski sustavi za rad u stvarnom vremenu

- Nisu nužno najbrži, ali garantiraju odziv unutar nekih granica UVIJEK!
- Primjer:

Tablica 10.1. Izmjerena kašnjenja do početka obrade prekida u μs ([Laurich, 2004])

	2.4 Linux	2.6 Linux	2.4 Linux RTAI	2.4 Linux LXRT
prosječna kašnjenja	12 – 14	12 – 14	8 – 10	10 – 12
najveća kašnjenja	4446	578	42	50

- RTAI i LXRT imaju zasebnu jezgru za SRSV i zato su brži i predvidljiviji



Legenda: svaki znak "+" označava jedan događaj u običnom Linuxu dok znak "×" događaj u prilagođenom Linuxu (SRSV inačici); ordinata predstavlja vrijeme odziva, a apscisa redni broj događaja

Slika 10.3. Usporedba vremena odziva na Linuxu i njegovoj prilagođenoj inačici ([Clark, 2008])

10.3.2. Operacijski sustavi opće namjene

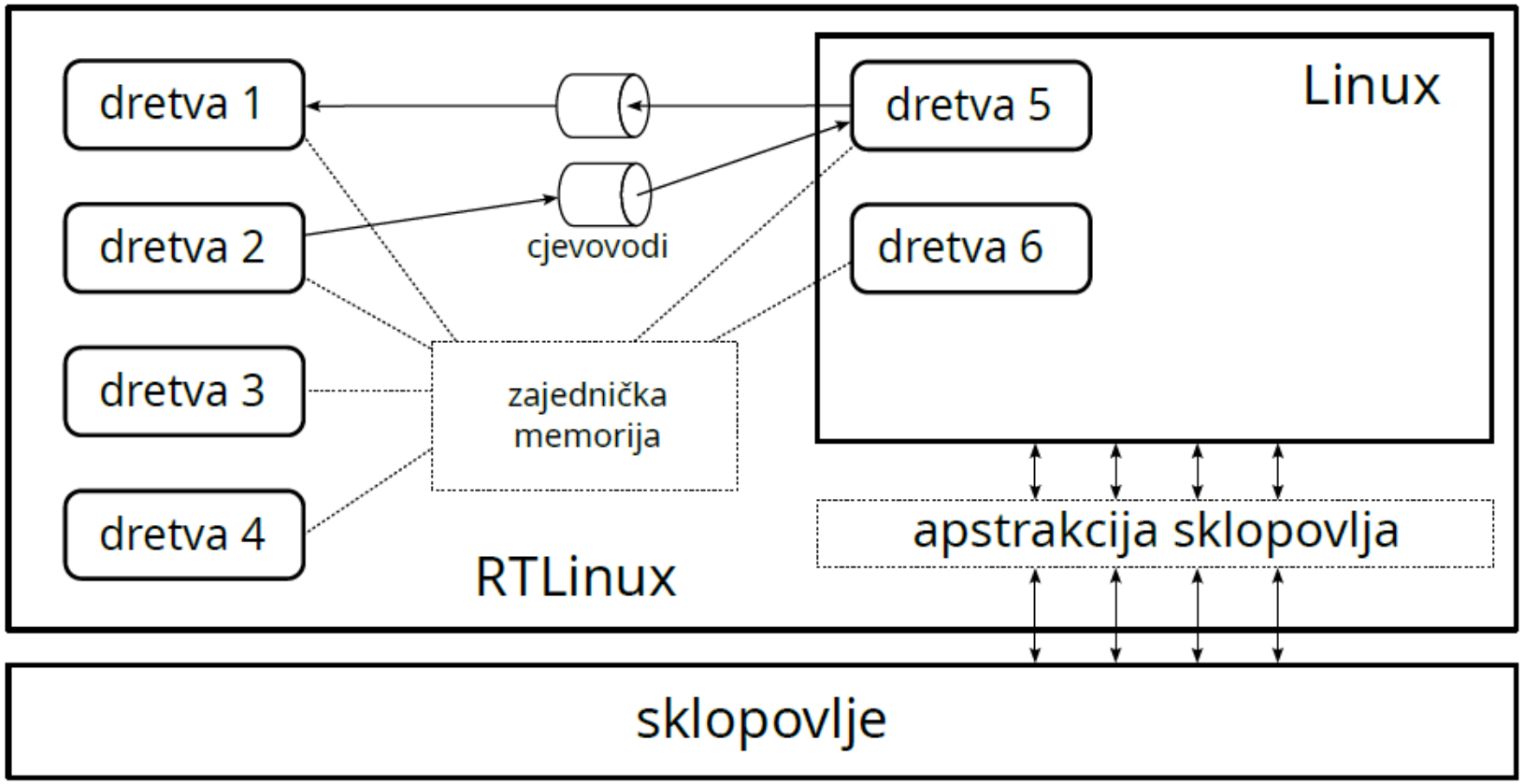
- loša svojstva u kontekstu SRSV-a
  - vrijeme odziva
  - deterministično ponašanje
  - zahtjevi prema sklopovlju
- ali:
  - dostupni mnogi programi, alati, podržani sustavi
  - mnogo programera zna za njih programirati
  - podrška u budućnosti
- možda se ipak za neke nekritične sustave može i on iskoristiti
- ili možda se taj OS može podesiti da bude nešto bolji?
  - npr. maknuti/zaustaviti nepotrebne usluge, ...

10.3.3. Prilagođeni operacijski sustavi opće namjene

- Prilagođeni = ponešto izmijenjeni u jezgri, ne samo postavkama
  - Primjeri: Real-Time Linux, RTLinux, Windows 10 IoT Core
- **Real-Time Linux** (CONFIG_PREEMPT_RT patch)
  - običan Linux uz primjenu dodatka CONFIG_PREEMPT_RT na kod jezgre
  - Linux ima monolitnu jezgru – jezgrine funkcije su „veće”, duže traju
  - dugo trajanje nekih jezgrinih funkcija ili nekih njenih dijelova znači da se novi zahtjevi za prekid neće prihvaćati za to vrijeme (u jednoprocesorskim sustavima)
  - dodatak mijenja neke jezgrine funkcije da se mogu prekinuti i prije kraja, dodaje „točke prekida” unutra (a da sve ostane konzistentno)
  - od rujna 2024., inačice jezgre 6.12. taj dodatak je integriran u izvorni kod Linuxa, u budućnosti će se automatski ažurirati s jezgrom Linuxa
    - lakše aktiviranje te mogućnosti: pri prevođenju samo postaviti zastavicu CONFIG_PREEMPT_RT

RTLinux

- zasebna jezgra za upravljanje
- Linux je „samo” proces u takvom sustavu („korisničko sučelje”)



10.3.4. Odabir operacijskih sustava

- odabir ovisi o zahtjevima
- nekad nemamo baš odabira, moramo odabrati RTOS
- ipak češće se može odabrati i nešto ne toliko „dobro” (skupo)
- najčešće treba napraviti kompromis uzimajući u obzir:
  - potrebna vremenska svojstva
  - podrška sklopovlju
  - dostupnost programera
  - dostupnost korisničke podrške
  - cijena
    - vrlo često je najjeftinije „kupiti”
    - vlastiti razvoj može potrajati, imati greške, ...