P9

Uvod u RTOS

RTOS

-RT

-OS

RTOS

-RT

Definicija:

Sistem za rad u realnom vremenu ima vremenska ograničenja na realizaciju odziva koja ako se premaše povećava se rizik od neadekvatnog rada ili otkaza sistema.

-OS

Klasifikacija RTOS

Odziv koji je duži od zadatih vremenskih ograničenja na odziv dovodi kod:

-*Hard RTOS* do otkaza sistema. Tolerancija na ovakav odziv je minimalna ili nikakava.

-*Firm RTOS* do tolerisanog neadekvatnog odziva ukoliko je verovatnoća pojave dugačkog odziva mala.

-*Soft RTOS* do smanjenja perfomansi sistema, ali to nije kritično i dozvoljeno je.

Primeri

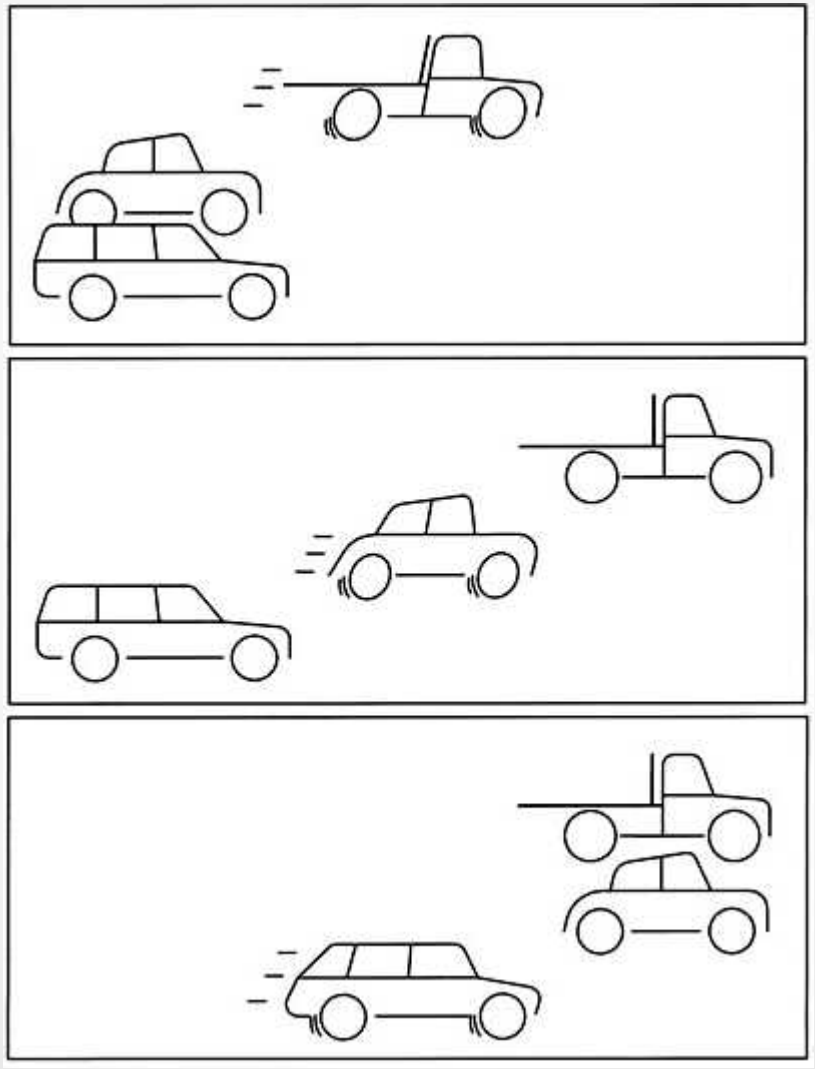
-*Hard RTOS:* navigacija helikoptera, navigacija raketa, robotizovana hirurgija itd. (potencijalno katastrofalni scenariji u slučaju otkaza)

-*Firm RTOS:* navigacija robota koji uklanja korove u poljoprivredi (uništiće možda deo useva, ali nije katastrofalni scenario)

-*Soft RTOS:* govorni automat, pokretne stepenice itd. (smanjenje perfomansi sistema, ali nije kritično)

Multitasking RTOS

Multitasking?
3 vozila, a 1 prenosivi motor



Multitasking

Definicija multitaskinga:
Sposobnost izvršavanja višestrukih taskova na *pseudo* simultani način

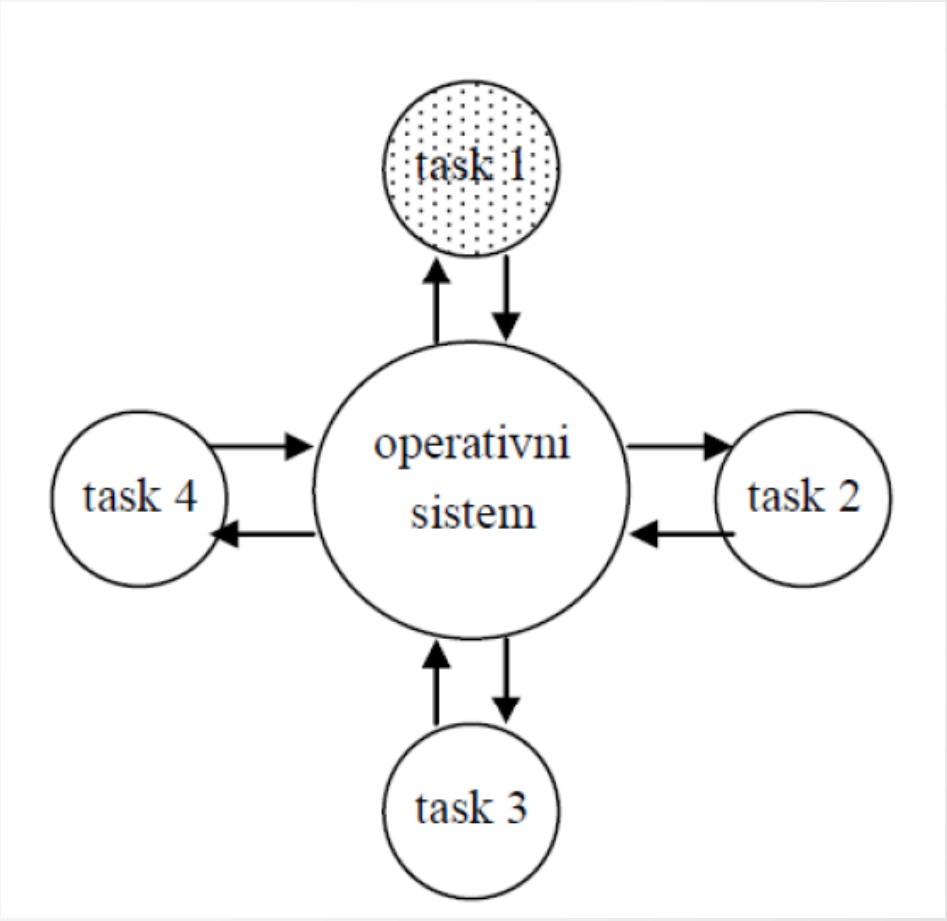
Multitasking

1)Taskovi
2)Context switching
3)Komunikacija među taskovima
4)Upravljanje prioritetima taskova
5)Kontrola vremena

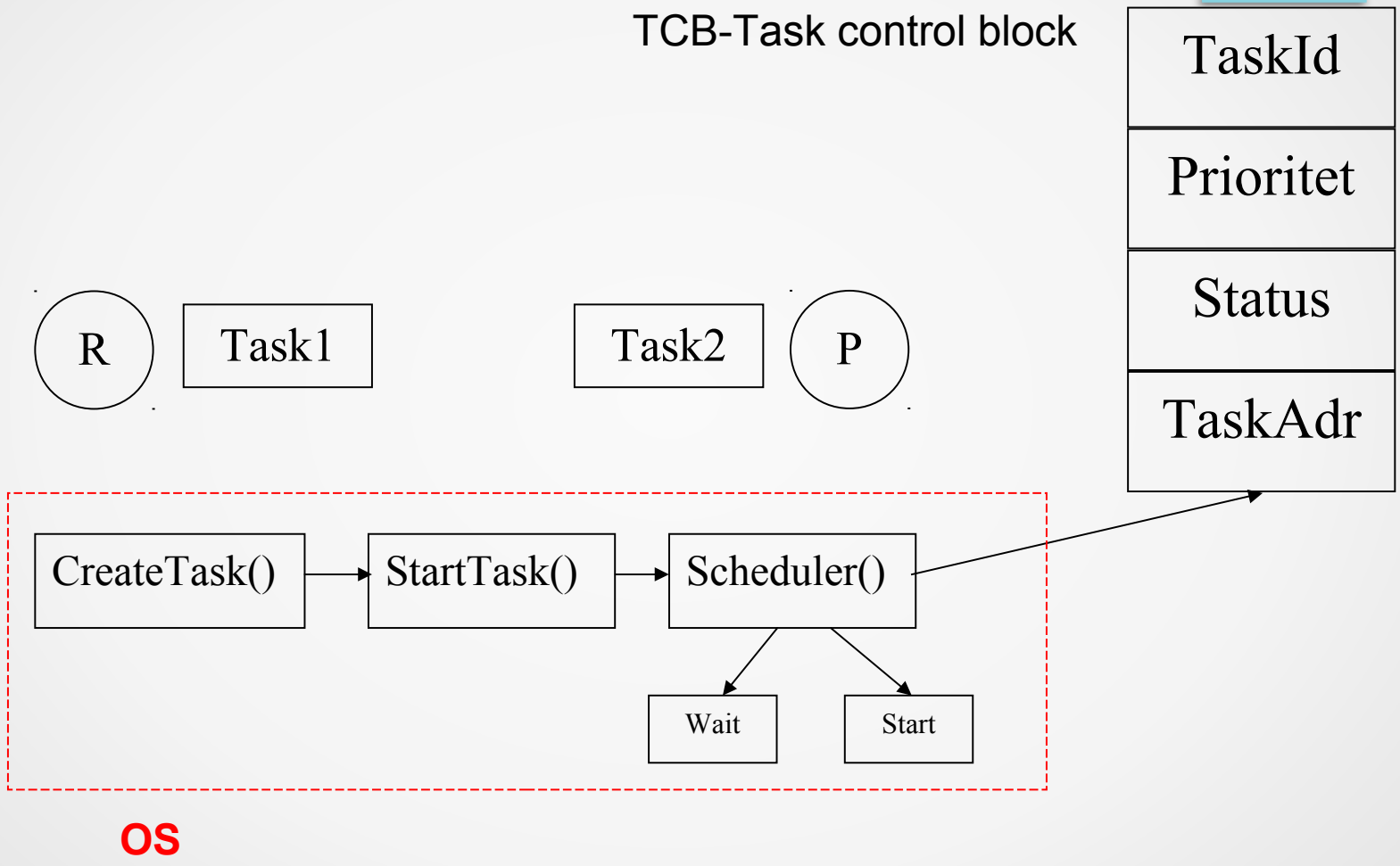
Multitasking RTOS

-Podela zadatka na taskove

-Princip multipleksa



Primer sa 2 taska



RTOS.h

```
#define TaskId_1 10
#define TaskId_2 20

struct TaskTabela
{
    void (*ptrTask) (void);//adresa taska
    int Prioritet; //prioritet
    int Ready;//stanje taska (spreman, čeka, zauzet i sl.)
    int TaskId;//id
}arrTaskTable[3];

int iTaskcount;

int PriorityTable[3];
int iHighPriorityTask;
int iIndexPriority;

int iWait;

void StartOS(void);
void Scheduler(void);
void CreateTask(void (*ptrTask)(void), int iPriority, int iTaskId );
void StartTask(int iTaskId);
void WaitTask();
void ContextSwitch(void);

void Task1(void);
void Task2(void);
```

TestRTOS.c

```c
int  main(void)
{
    printf("Kreiranje Taskova \n");

    createTask(Task1, 1, TaskId_1);
    createTask(Task2, 2, TaskId_2);

    //Startuj OS
    StartOS();

    return 0;
}

void Task1()
{
    while(1)
    {
        printf("Ovo je Task 1 \n");

        //wait
        Sleep(1000);//1 sec

        //Stop the task
        WaitTask();

    }
}
```

```c
void Task2()
{
    while(1)
    {
        printf("Ovo je Task 2 \n");

        //wait
        Sleep(1000);//1 sec

        //start the task
        StartTask(TaskId_1);
    }
}
```

RTOS.c

```
void StartOS()
{
    //pozovi Scheduler
    Scheduler();
}
 void Scheduler()
{
    int iIndex;

    //Izaberi task sa najvecim prioritetom

    iHighPriorityTask = 10;

    for (iIndex = 0; iIndex < iTaskcount ; iIndex++)
    {
        if ((arrTaskTable[iIndex].Priority <= iHighPriorityTask) &&
                (arrTaskTable[iIndex].Ready ==1 ))
        {
            iHighPriorityTask = arrTaskTable[iIndex].Priority;
            iIndexPriority = iIndex;
        }
    }

    //Aktiviraj task sa najvecim prioritetom
    (*arrTaskTable[iIndexPriority].ptrTask)();

}
```

RTOS.c

```
void CreateTask(void (*ptrTask)(void), int iPriority, int iTaskId )
{
    //Postavi prioritet
    arrTaskTable[iTaskcount].Priority = iPriority;

    //Adresa taska
    arrTaskTable[iTaskcount].ptrTask = ptrTask;

    //Task ID
    arrTaskTable[iTaskcount].TaskId = iTaskId;

    //Task je spreman
    arrTaskTable[iTaskcount].Ready = 1;

    //Uvecaj brojac taskova
    iTaskcount ++;
}

void waitTask()
{
    // Task je zauzet
    arrTaskTable[iIndexPriority].Ready = 0;

    //Pozovi Scheduler
    Scheduler();
}
```

```
void startTask(int TaskId)
{
    int iIndex;

    // pronadji task koji je spreman
    for (iIndex = 0; iIndex <iTaskcount; iIndex ++ )
    {
        if (TaskId == arrTaskTable[iIndex].TaskId)
        {
            arrTaskTable[iIndex].Ready = 1;
        }
    }

    //Pozovi Scheduler
    Scheduler();
}
```

Vrste RTOS

Zavisno od načina promene aktivnog taska imamo:
-**Kooperativni** RTOS

-**Preemptive** RTOS

-**FSM** RTOS

Kod kooperativnih RTOS se promena aktivnog taska vrši iz tekućeg taska pozivom funkcije OS.

Kod preemptive RTOS se promena aktivnog taska vrši forsirano, od strane OS, inicirano hardverskim/softverskim događajem ili vremenskim intervalom.

Primer kooperativnog RTOS sa 3 taska

```
void task1(void)
{
  while(1)
  {
    OS_Switch();
    ....
  }
}
```

```
void task2(void)
{
  while(1)
  {
    OS_Switch();
    ....
  }
}
```

```
void task3(void)
{
  while(1)
  {
    OS_Switch();
    ....
  }
}
```

Stanja taska

Informacije o trenutnom stanju taska mogu biti:

- Aktivan (*Active*) – task je aktivan (upravo se izvršava)
- Spreman (*Ready*) – task nije aktivan, ali je spreman za rad i može se aktivirati.
- U čekanju (*Wait*) – task je u stanju čekanja na neki događaj ili istek vremenskog intervala i nije spreman za aktiviranje.
- Suspendovan (*Suspended*) – task je privremeno suspendovan i ignoriše se sve do ukidanja ovog stanja.

Prelaz

Prelaz iz jednog taska u drugi:

- OS_Switch()
- Mora task biti *Ready*
- Prioritet
- *Round Robin*

Problem steka

```
        void Task1(void)                    void Task2(void)
        {                                   {
          while (1)                           while (1)
          {                                   {
            ...                                 ...
A1          OS_Switch();            A2          OS_Switch();
            ...                                 ...
B1          func_t1();              B2          func_t2();
C1          ....                    C2          ....
          }                                   }
        }                                   }

        void func_t1(void)                  void func_t2(void)
        {                                   {
          ....                                ....
D1        OS_Switch();              D2        OS_Switch();
          ....                                ....
        }                                   }
```

Stek	
	...
S3	ret_t2
S2	ret_t1
S1	SP
	...

Rešenja

Rešenja problema steka:
1-poziv funkcije za promenu taska isključivo iz glavne funkcije taska

2-u slučaju da svaki task ima sopstveni stek može i iz podfunkcija

Deadlock

```
void Task1(void)
{
  ...
  while (1)
  {
    ...
    OS_WaitSem(ev_1,0);
    OSSetSem(ev_2);
    ...
  }
}
```

```
void Task2(void)
{
  ...
  while (1)
  {
    ...
    OS_WaitSem(ev_2,0);
    OSSetSem(ev_1);
    ...
  }
}
```

Primer 2 - Femtos

```
void main(void)
{
  OSInit(0);
  OSCreateTask(Task1);
  OSCreateTask(Task2);
  OSCreateTask(Task3);
  OSRun();
}
```

```
void Task1(void)
{
  static char led;
  while (1)
  {
    RED = ~led;
    OS_Sleep(250);
    led++;
  }
}
```

```
void Task2(void)
{
  static char led;
  while (1)
  {
    YELLOW = ~led;
    OS_Sleep(200);
    led++;
  }
}
```

```
void Task3(void)
{
  static char led;
  while (1)
  {
    GREEN = ~led;
    OS_Sleep(150);
    led++;
  }
}
```

Primer 3 – ArdOS – kompakt RTOS

- Kernel (bez taskova): ispod 2k flash-a i manje od 200 B RAM-a
- Preemptive scheduling sa prioritetima
- Kooperativni scheduling
- Sleep funkcija koja oslobađa MCU za druge taskove
- Binarni i brojački semafori
- Mutex
- FIFO i prioritetni messaging
- Konfigurabilan radi štednje memorije

Prednosti/mane

U odnosu na softverske FSM:

1-lakše pisanje programa (ali paziti na međuzavisnosti taskova)

2-veći izlazni kod

3-eventualni problemi sa stekom

HVALA NA PAŽNJI !

```
CreateTask(HVALA_NA_PAŽNJI!)
```