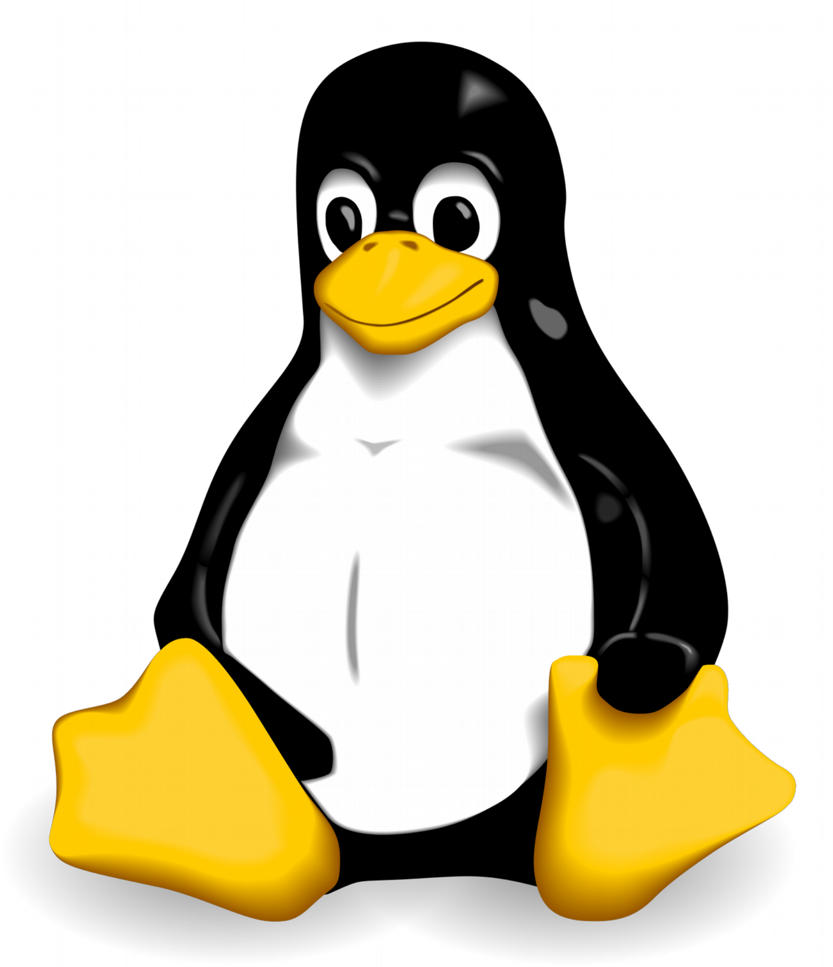
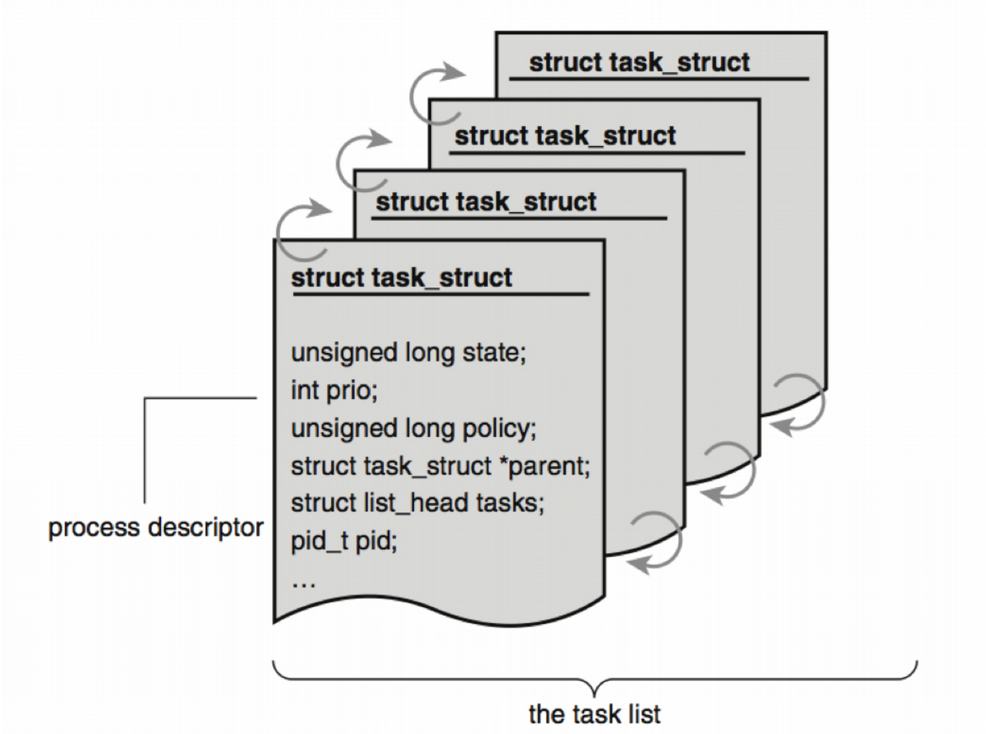
Modifikacija i *rebuild*-ovanje Linux kernela

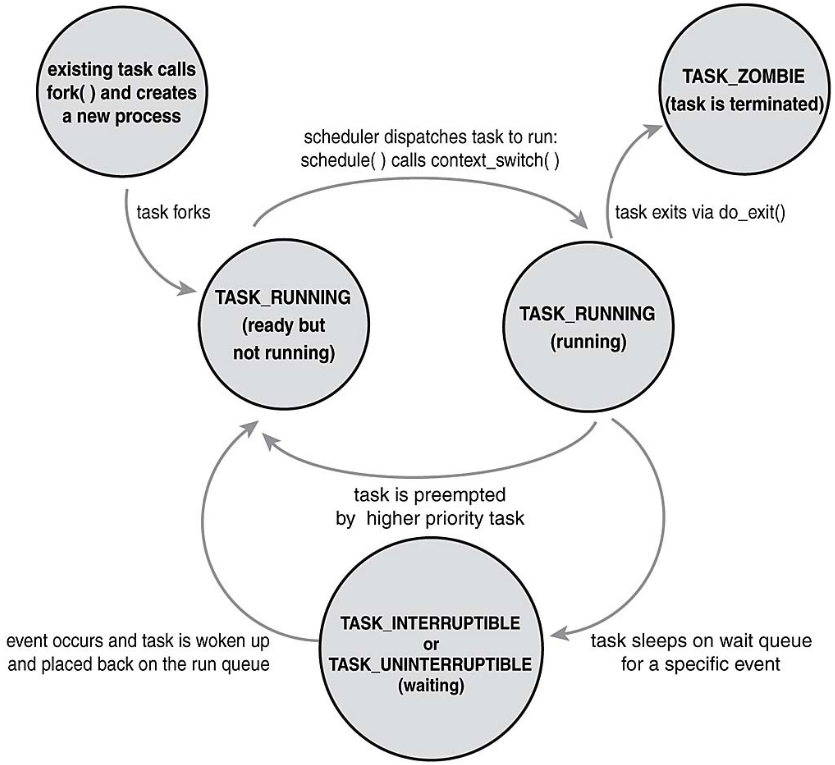


Aleksa Trajković	370
Filip Stamenković	342

Uvod

- Upravljanje procesima je jedno od glavnih zaduženja svakog operativnog sistema.
- CPU je jedan od najznačajnijih resursa u računarskom sistemu, koji konkurentni procesi moraju efikasno deliti.

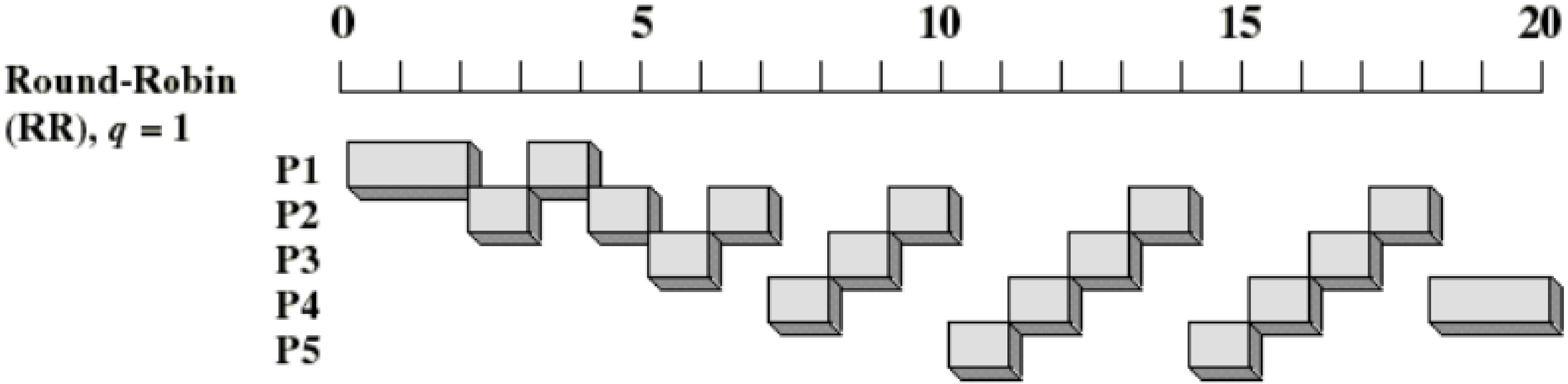



existing task calls fork( ) and creates a new process
TASK_ZOMBIE (task is terminated)
scheduler dispatches task to run: schedule( ) calls context_switch( )
task forks
task exits via do_exit()
TASK_RUNNING (ready but not running)
TASK_RUNNING (running)
task is preempted by higher priority task
event occurs and task is woken up and placed back on the run queue
TASK_INTERRUPTIBLE or TASK_UNINTERRUPTIBLE (waiting)
task sleeps on wait queue for a specific event

Process Scheduling

- Raspoređivanje (planiranje procesa, *process scheduling*) je ključno za multiprogramiranje.
- Cilj raspoređivanja je da se procesoru dodeljuju procesi za izvršenje, na način da se zadovolje zahtevi sistema, kao što su vreme odziva i propusna moć.
- Postoji puno algoritama za raspoređivanje u literaturi, jedan od najjednostavnijih je algoritam kružnog raspoređivanja (*round-robin*).

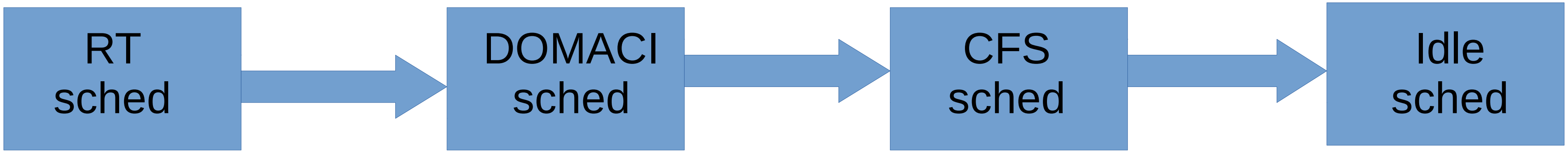
Round-Robin (RR)



- **Funkcija selekcije:** ista kao kod FCFS
- **Mod odluke:** sa prekidanjem (*preemptive*)
  - Procesu se dopušta da se izvršava dok mu ne istekne dodeljeni vremenski period (kvant tipično 10 do 100 ms)
  - Tada se javlja prekidni signal od časovnika i proces se vraća u red spremnih procesa

Cilj domaćeg zadatka

- Dodavanje novog *scheduler*-a (“domaći”) u red već postojećih.
- Dodavanje sistemskog poziva za prikupljanje statistike o procesima koji su se izvršili na novom *scheduler*-u.

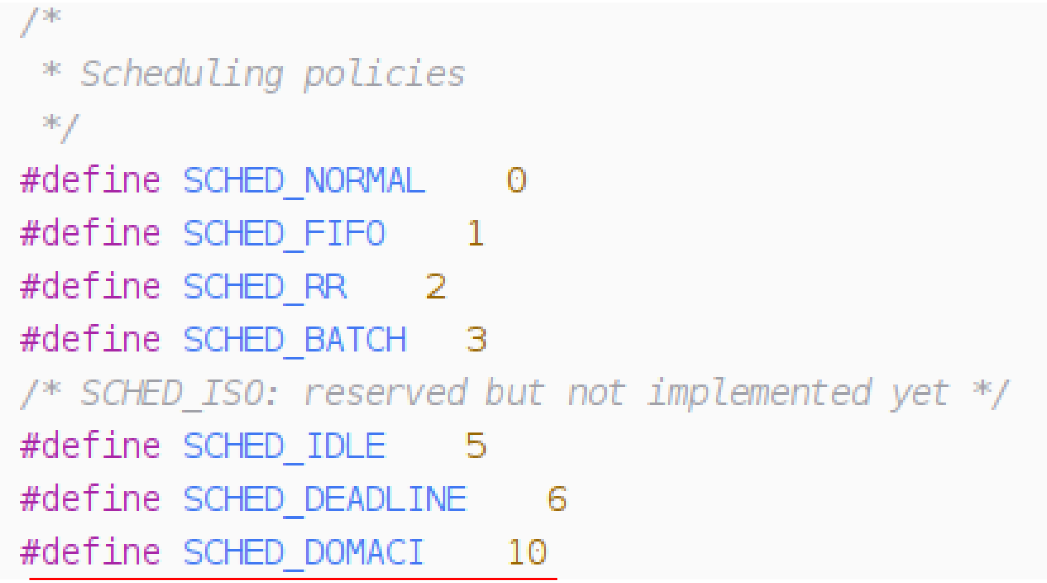


Linux Scheduler

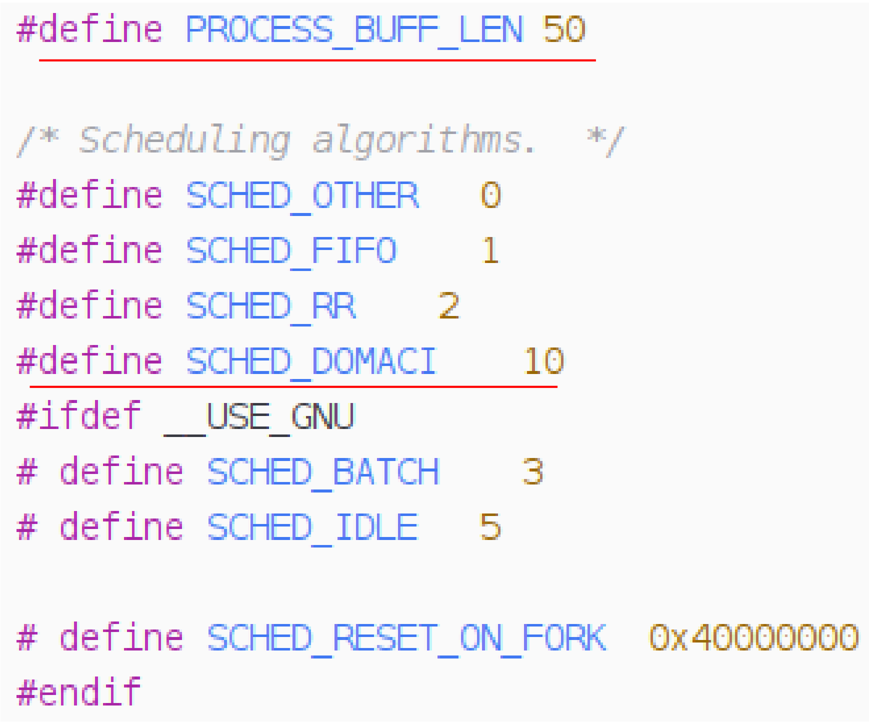
- *Linux scheduler* je modularan i omogućava da se različitim tipovima procesa dodele različiti algoritmi raspoređivanja.
- Svaki *scheduling class* enkapsulira *scheduling policy*.
  - Real-Time Scheduler
  - Completely Fair Scheduler
  - Idle Scheduler
  - …
- Scheduling klase su hijerarhijski organizovane.

Scheduling policy

- Prvi korak u implementaciji novog scheduler-a je definisati scheduling policy.
  - ***(kernel-source)/include/uapi/linux/sched.h***
  - ***/usr/include/bits/sched.h***



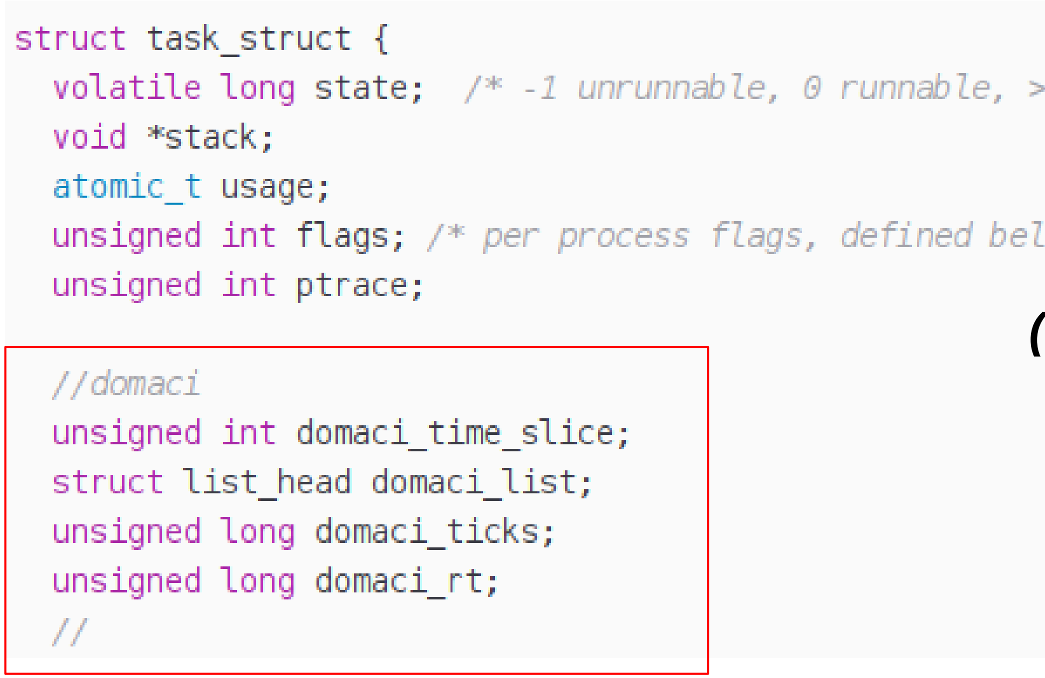
```
/*
 * Scheduling policies
 */
#define SCHED_NORMAL    0
#define SCHED_FIFO    1
#define SCHED_RR    2
#define SCHED_BATCH   3
/* SCHED_ISO: reserved but not implemented yet */
#define SCHED_IDLE    5
#define SCHED_DEADLINE    6
#define SCHED_DOMACI    10
```



```
#define PROCESS_BUFF_LEN 50

/* Scheduling algorithms.  */
#define SCHED_OTHER   0
#define SCHED_FIFO    1
#define SCHED_RR    2
#define SCHED_DOMACI    10
#ifdef __USE_GNU
# define SCHED_BATCH    3
# define SCHED_IDLE   5

# define SCHED_RESET_ON_FORK  0x40000000
#endif
```

Dve najvažnije strukture, ***task_struct*** i ***rq***

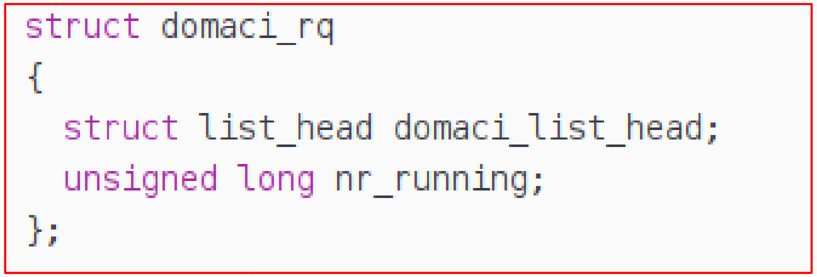
(kernel-source)/include/linux/sched.h



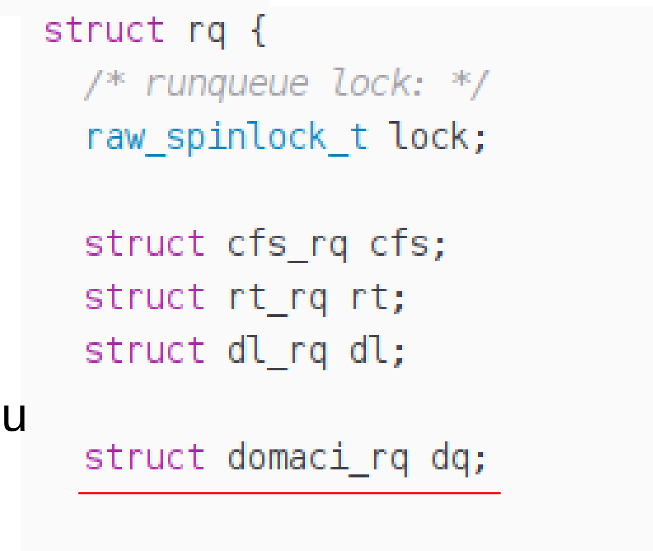
```
struct task_struct {
  volatile long state;  /* -1 unrunnable, 0 runnable, >0 stopped */
  void *stack;
  atomic_t usage;
  unsigned int flags; /* per process flags, defined below */
  unsigned int ptrace;

  //domaci
  unsigned int domaci_time_slice;
  struct list_head domaci_list;
  unsigned long domaci_ticks;
  unsigned long domaci_rt;
  //
```

(dodat je i kod za inicijalizaciju dodatih polja strukture u ***/include/linux/init_task.h*** i ***/kernel/fork.c***)

(kernel-source)/kernel/sched/sched.h



```
struct domaci_rq
{
  struct list_head domaci_list_head;
  unsigned long nr_running;
};
```



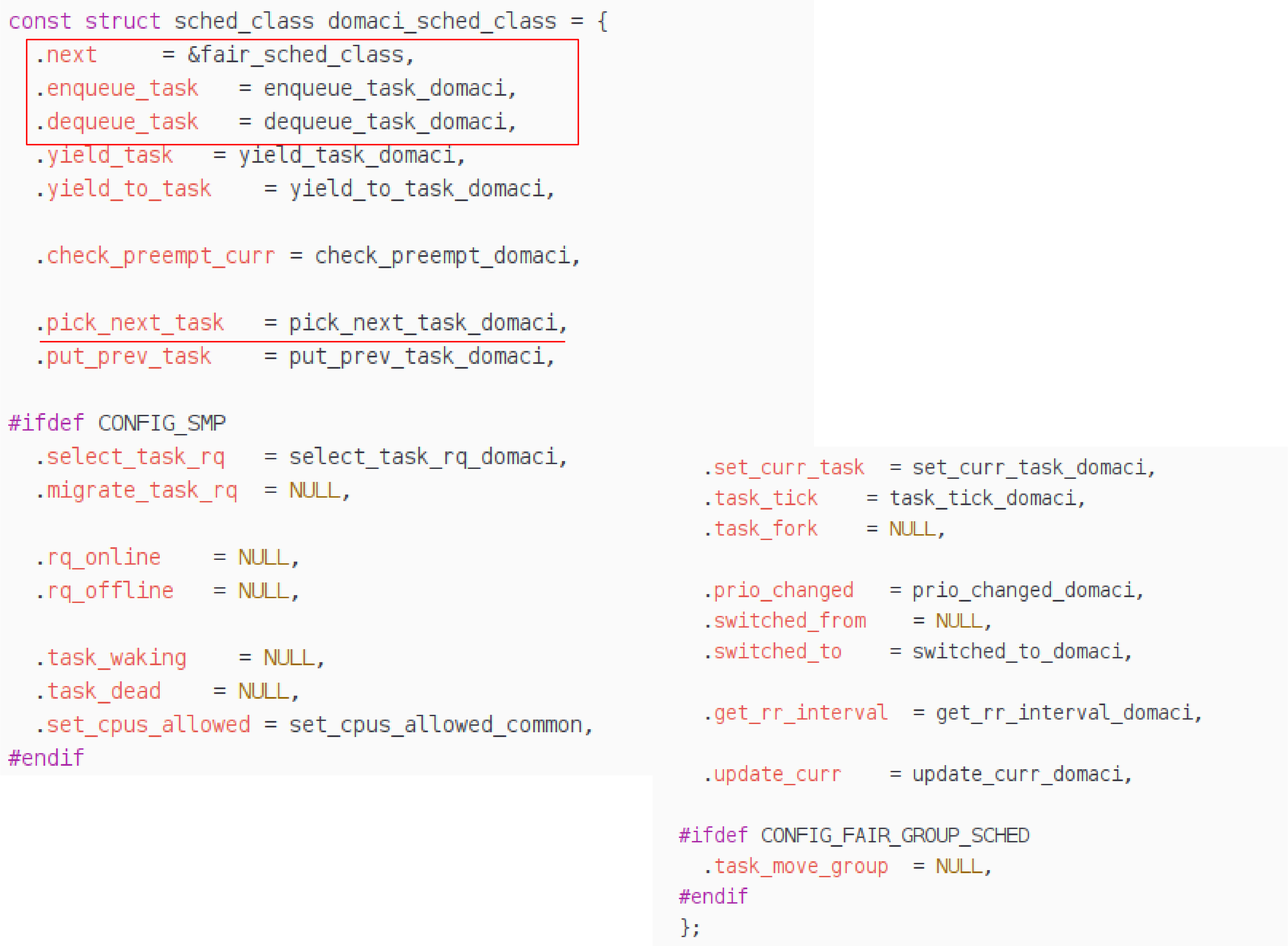
```
struct rq {
  /* runqueue lock: */
  raw_spinlock_t lock;

  struct cfs_rq cfs;
  struct rt_rq rt;
  struct dl_rq dl;

  struct domaci_rq dq;
```

(dodat je i kod za inicijalizaciju dodatih polja strukture u ***/kernel/sched/core.c***)

domaci scheduler class

- Opisan u fajlu /kernel/sched/domaci.c

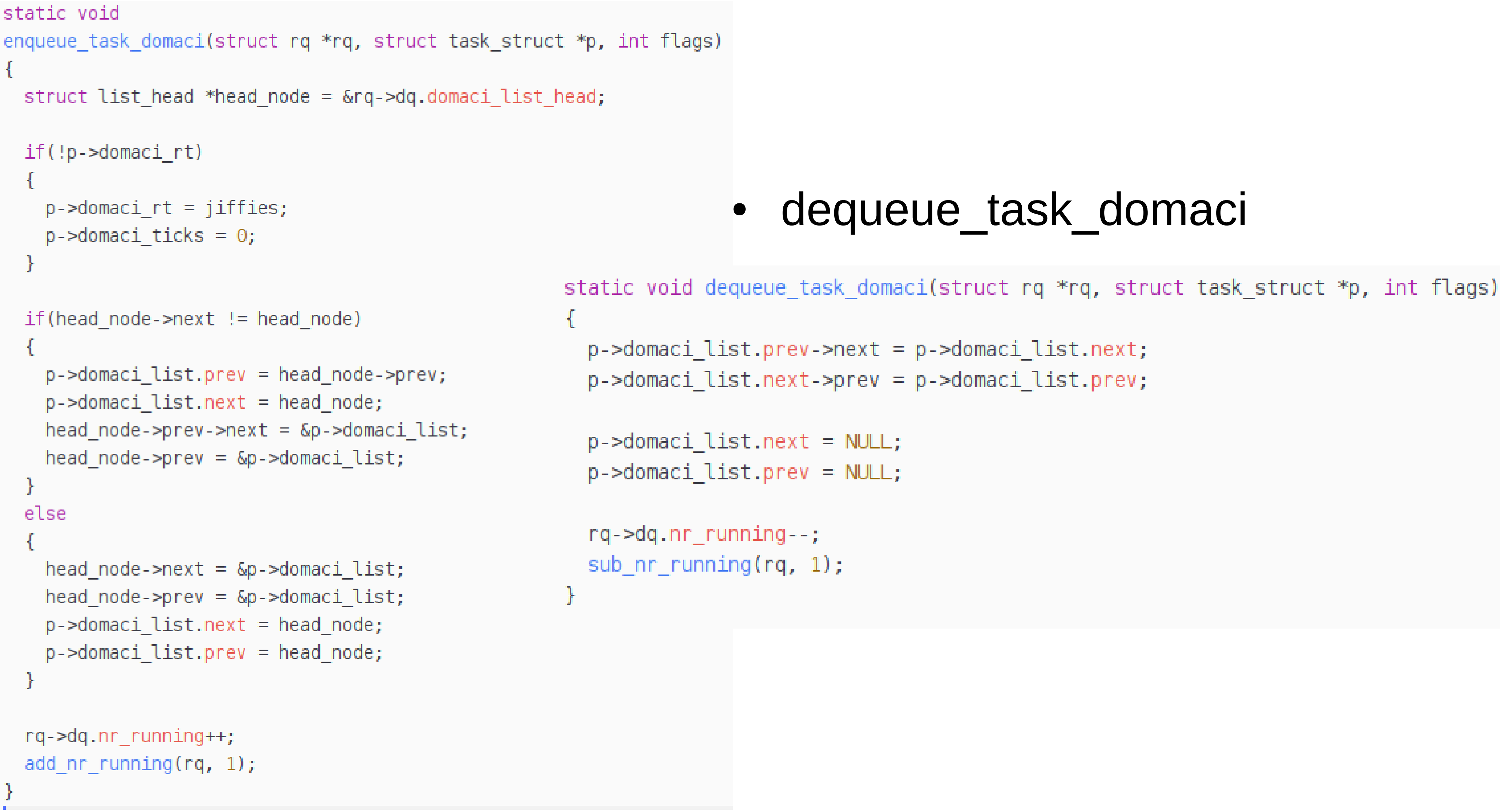
- Bitne funkcije:
  - enqueue_task_domaci()
  - dequeue_task_domaci()
  - pick_next_task_domaci()
  - task_tick_domaci()
  - ...



```c
const struct sched_class domaci_sched_class = {
  .next     = &fair_sched_class,
  .enqueue_task   = enqueue_task_domaci,
  .dequeue_task   = dequeue_task_domaci,
  .yield_task   = yield_task_domaci,
  .yield_to_task    = yield_to_task_domaci,

  .check_preempt_curr = check_preempt_domaci,

  .pick_next_task   = pick_next_task_domaci,
  .put_prev_task    = put_prev_task_domaci,

#ifdef CONFIG_SMP
  .select_task_rq   = select_task_rq_domaci,
  .migrate_task_rq  = NULL,

  .rq_online    = NULL,
  .rq_offline   = NULL,

  .task_waking    = NULL,
  .task_dead    = NULL,
  .set_cpus_allowed = set_cpus_allowed_common,
#endif
  .set_curr_task  = set_curr_task_domaci,
  .task_tick    = task_tick_domaci,
  .task_fork    = NULL,

  .prio_changed   = prio_changed_domaci,
  .switched_from    = NULL,
  .switched_to    = switched_to_domaci,

  .get_rr_interval  = get_rr_interval_domaci,

  .update_curr    = update_curr_domaci,

#ifdef CONFIG_FAIR_GROUP_SCHED
  .task_move_group  = NULL,
#endif
};
```

domaci scheduler

- enqueue_task_domaci



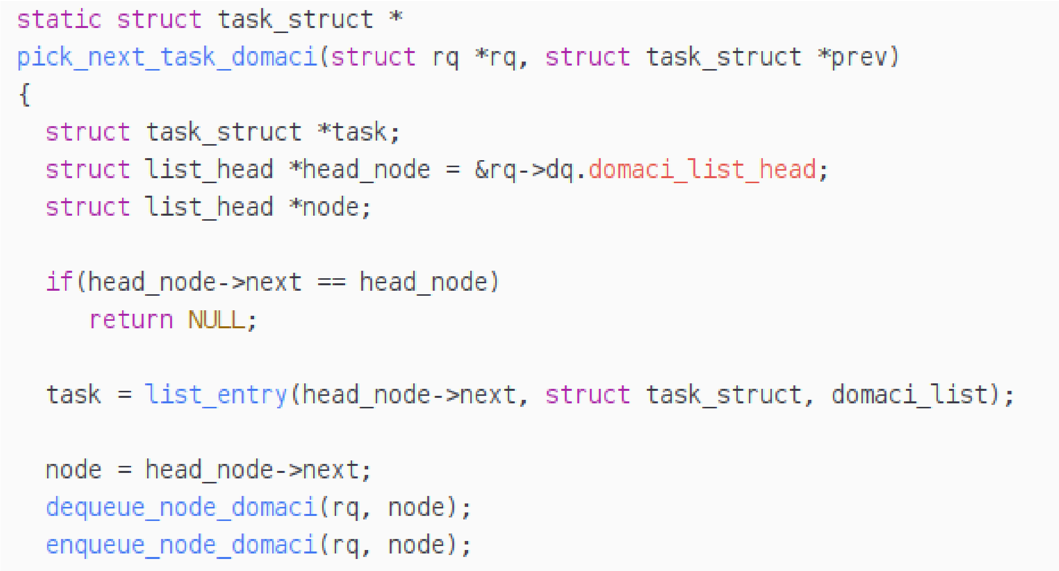
```
static void
enqueue_task_domaci(struct rq *rq, struct task_struct *p, int flags)
{
  struct list_head *head_node = &rq->dq.domaci_list_head;

  if(!p->domaci_rt)
  {
    p->domaci_rt = jiffies;
    p->domaci_ticks = 0;
  }

  if(head_node->next != head_node)
  {
    p->domaci_list.prev = head_node->prev;
    p->domaci_list.next = head_node;
    head_node->prev->next = &p->domaci_list;
    head_node->prev = &p->domaci_list;
  }
  else
  {
    head_node->next = &p->domaci_list;
    head_node->prev = &p->domaci_list;
    p->domaci_list.next = head_node;
    p->domaci_list.prev = head_node;
  }

  rq->dq.nr_running++;
  add_nr_running(rq, 1);
}
```

- dequeue_task_domaci

```
static void dequeue_task_domaci(struct rq *rq, struct task_struct *p, int flags)
{
  p->domaci_list.prev->next = p->domaci_list.next;
  p->domaci_list.next->prev = p->domaci_list.prev;

  p->domaci_list.next = NULL;
  p->domaci_list.prev = NULL;

  rq->dq.nr_running--;
  sub_nr_running(rq, 1);
}
```

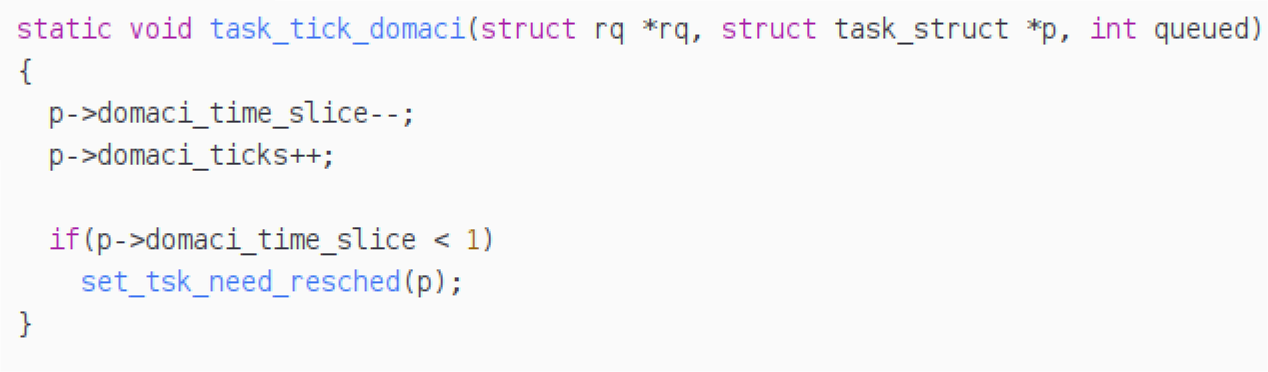
domaci scheduler

- pick_next_task_domaci



```c
static struct task_struct *
pick_next_task_domaci(struct rq *rq, struct task_struct *prev)
{
  struct task_struct *task;
  struct list_head *head_node = &rq->dq.domaci_list_head;
  struct list_head *node;

  if(head_node->next == head_node)
    return NULL;

  task = list_entry(head_node->next, struct task_struct, domaci_list);

  node = head_node->next;
  dequeue_node_domaci(rq, node);
  enqueue_node_domaci(rq, node);

  task->domaci_time_slice = DOMACI_SLICE;
  return task;
}
```

- task_tick_domaci



```c
static void task_tick_domaci(struct rq *rq, struct task_struct *p, int queued)
{
  p->domaci_time_slice--;
  p->domaci_ticks++;

  if(p->domaci_time_slice < 1)
    set_tsk_need_resched(p);
}
```

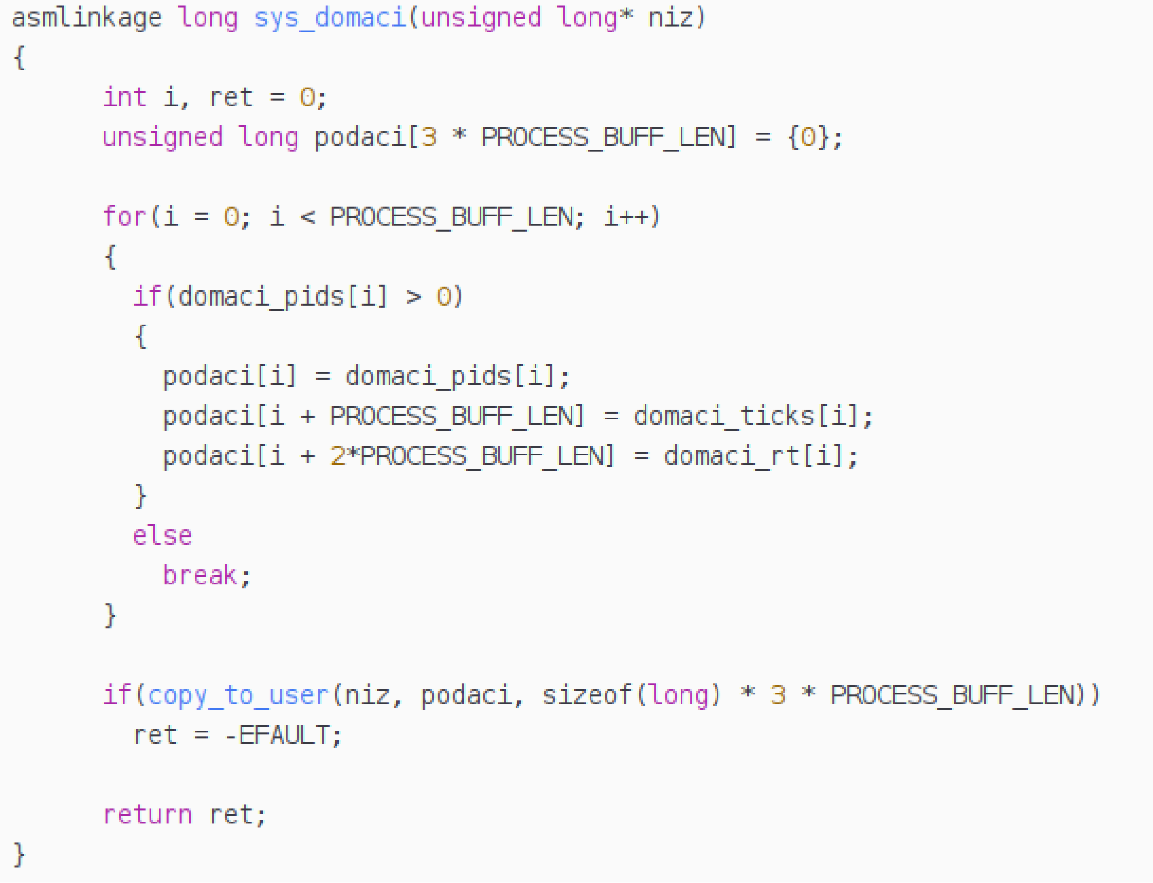
Sistemski poziv

- Sistemski poziv *sys_domaci* dodat u *domaci* scheduler
- Prikazivanje statistike procesa koji su se izvršili na *domaci* sched

- Statistika obuhvata:
  - pid procesa
  - vreme izvršenja na CPU
  - vreme provedeno od kreiranja do završenja procesa
- Funkcije:
  - sys_domaci - sistemski poziv
  - domaci_update_statistics - računanje statistike

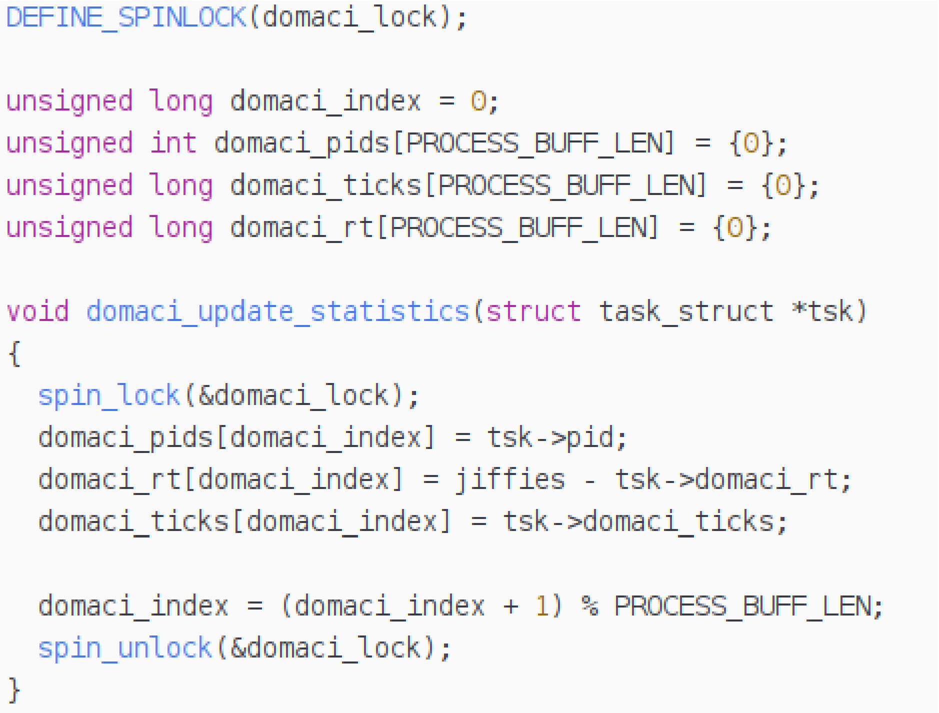
Sistemski poziv

- sys_domaci



```
asmlinkage long sys_domaci(unsigned long* niz)
{
      int i, ret = 0;
      unsigned long podaci[3 * PROCESS_BUFF_LEN] = {0};

      for(i = 0; i < PROCESS_BUFF_LEN; i++)
      {
        if(domaci_pids[i] > 0)
        {
          podaci[i] = domaci_pids[i];
          podaci[i + PROCESS_BUFF_LEN] = domaci_ticks[i];
          podaci[i + 2*PROCESS_BUFF_LEN] = domaci_rt[i];
        }
        else
          break;
      }

      if(copy_to_user(niz, podaci, sizeof(long) * 3 * PROCESS_BUFF_LEN))
        ret = -EFAULT;

      return ret;
}
```

- domaci_update_statistics



```
DEFINE_SPINLOCK(domaci_lock);

unsigned long domaci_index = 0;
unsigned int domaci_pids[PROCESS_BUFF_LEN] = {0};
unsigned long domaci_ticks[PROCESS_BUFF_LEN] = {0};
unsigned long domaci_rt[PROCESS_BUFF_LEN] = {0};

void domaci_update_statistics(struct task_struct *tsk)
{
  spin_lock(&domaci_lock);
  domaci_pids[domaci_index] = tsk->pid;
  domaci_rt[domaci_index] = jiffies - tsk->domaci_rt;
  domaci_ticks[domaci_index] = tsk->domaci_ticks;

  domaci_index = (domaci_index + 1) % PROCESS_BUFF_LEN;
  spin_unlock(&domaci_lock);
}
```

Sistemski poziv

- Dodavanje sistemskog poziva u tabelu sistemskih poziva
  - ***400 common domaci   sys_domaci*** - 64 bit (syscall_64.tbl)
  - ***400 i386 domaci   sys_domaci***   - 32 bit (syscall_32.tbl)

- Dodavanje sistemskog poziva **sys_domaci** u header fajl sistemskih poziva (/include/linux/syscalls.h)
  - asmlinkage long sys_domaci(unsigned long* niz);

asmlinkage - tag koji kompajler koristi da pokupi podatke sa steka, a ne iz registra.

Testiranje

- Operativni sistemi
  - Manjaro 15.12
  - Ubuntu 15.10
- Linux kernel
  - 4.4.1 (www.kernel.org)

Koraci prilikom build-ovanja

- Dodavanje *domaci.o* u odgovarajući Makefile (/kernel/sched/Makefile)
  - obj-y += idle_task.o fair.o **domaci.o** rt.o deadline.o stop_task.o

- Manjaro build:

```
$ make defconfig O=~/linux/linux-build/
$ make -jN CC=”ccache gcc”
O=~/linux/linux-build/ > /dev/null
```

- Instalacija:

```
$ cd ../linux-build
$ su root
# make modules_install
# cp -v arch/x86/boot/bzImage
        /boot/vmlinuz-4.4.1
# mkinitcpio -k 4.4.1
        -c /etc/mkinitcpio.conf
        -g /boot/initramfs-4.4.1.img
# update-grub
```

- Ubuntu build:

```
$ make defconfig
$ make -jN CC=”ccache gcc” deb-pkg
```

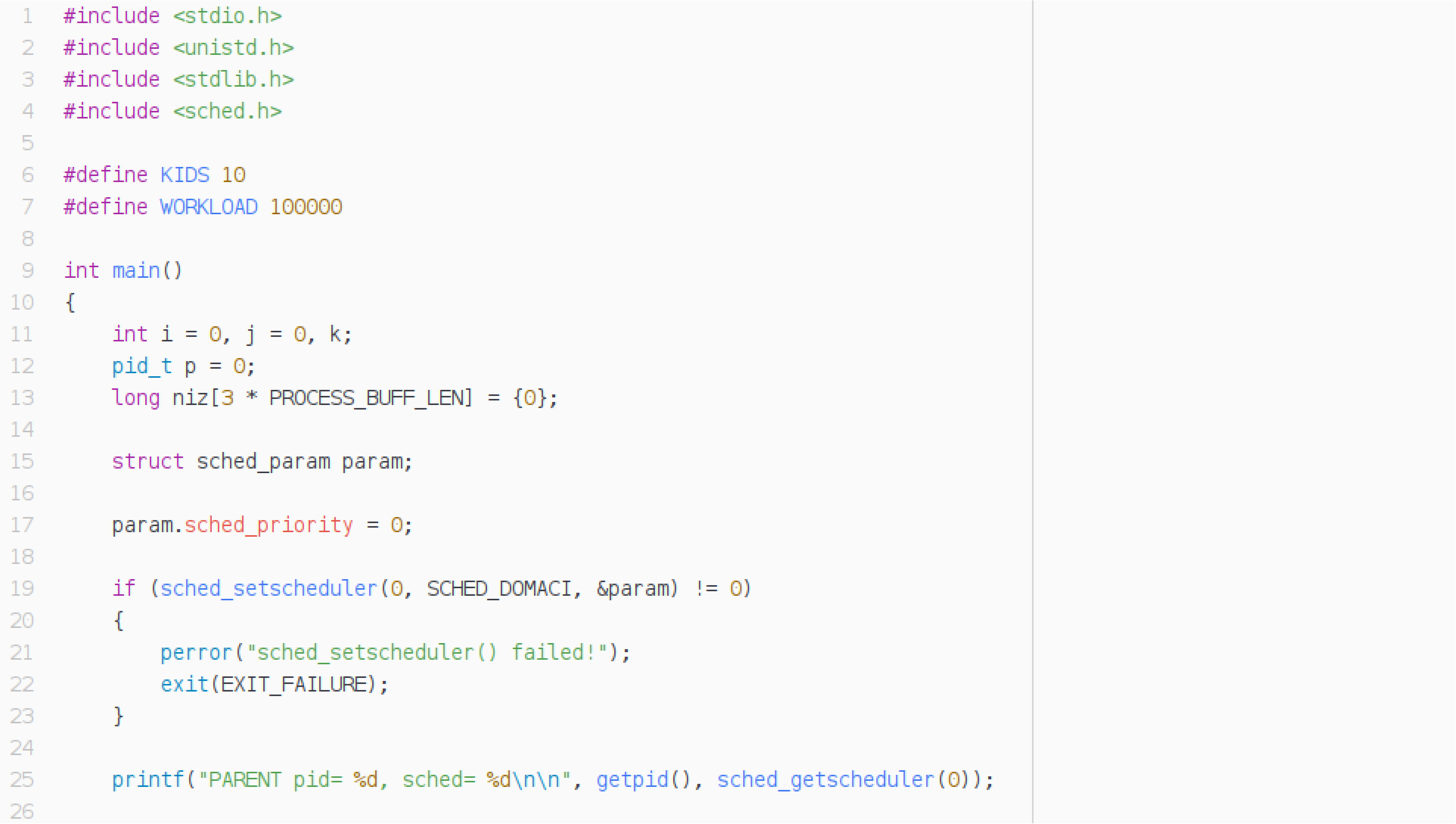
- Instalacija:

```
# dpkg -i linux-image-4.4.1.deb
# dpkg -i linux-headers-4.4.1.deb
```

Testiranje

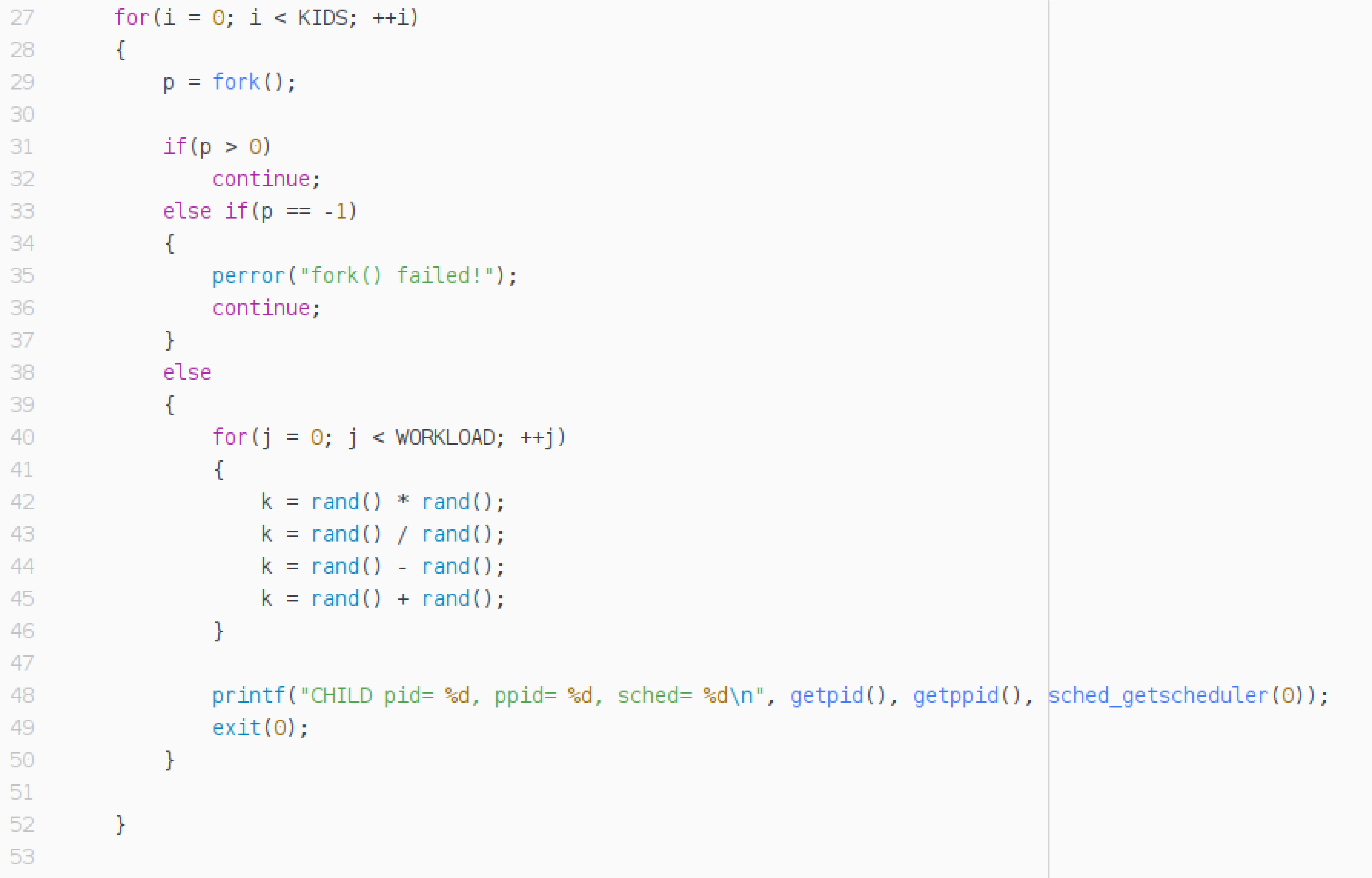
- Dodeliti procesu da se izvršava na *domaci* scheduleru, pomoću sistemskog poziva *setscheduler()*.
- Kreiranje 10 deteta procesa (pomoću *fork*()).
- Dodati svim procesima kompleksne zadatke.
- Pomoću sistemskog poziva *sys_domaci()*, dobaviti statistiku i odštampati je.

Korisnički program – part 1



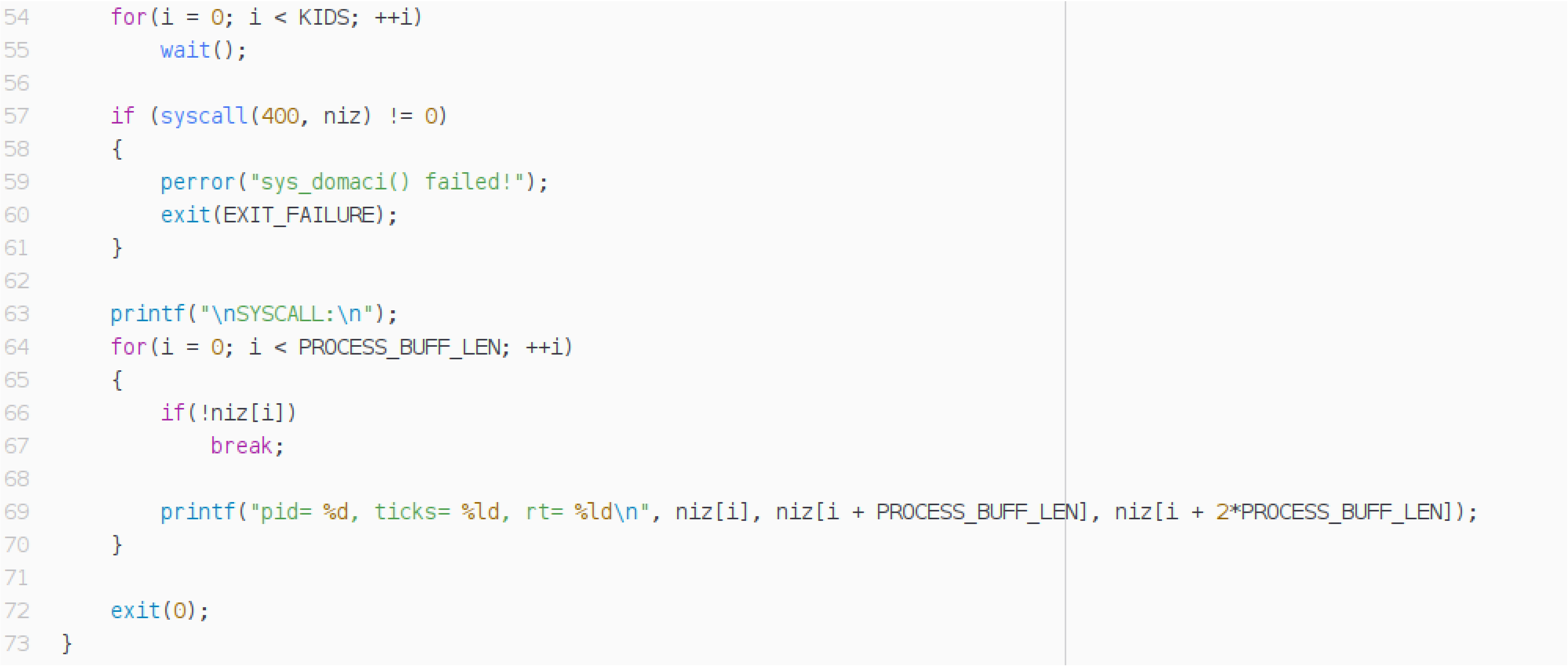
```c
#include <stdio.h>
#include <unistd.h>
#include <stdlib.h>
#include <sched.h>

#define KIDS 10
#define WORKLOAD 100000

int main()
{
    int i = 0, j = 0, k;
    pid_t p = 0;
    long niz[3 * PROCESS_BUFF_LEN] = {0};

    struct sched_param param;

    param.sched_priority = 0;

    if (sched_setscheduler(0, SCHED_DOMACI, &param) != 0)
    {
        perror("sched_setscheduler() failed!");
        exit(EXIT_FAILURE);
    }

    printf("PARENT pid= %d, sched= %d\n\n", getpid(), sched_getscheduler(0));

```

Korisnički program - part 2



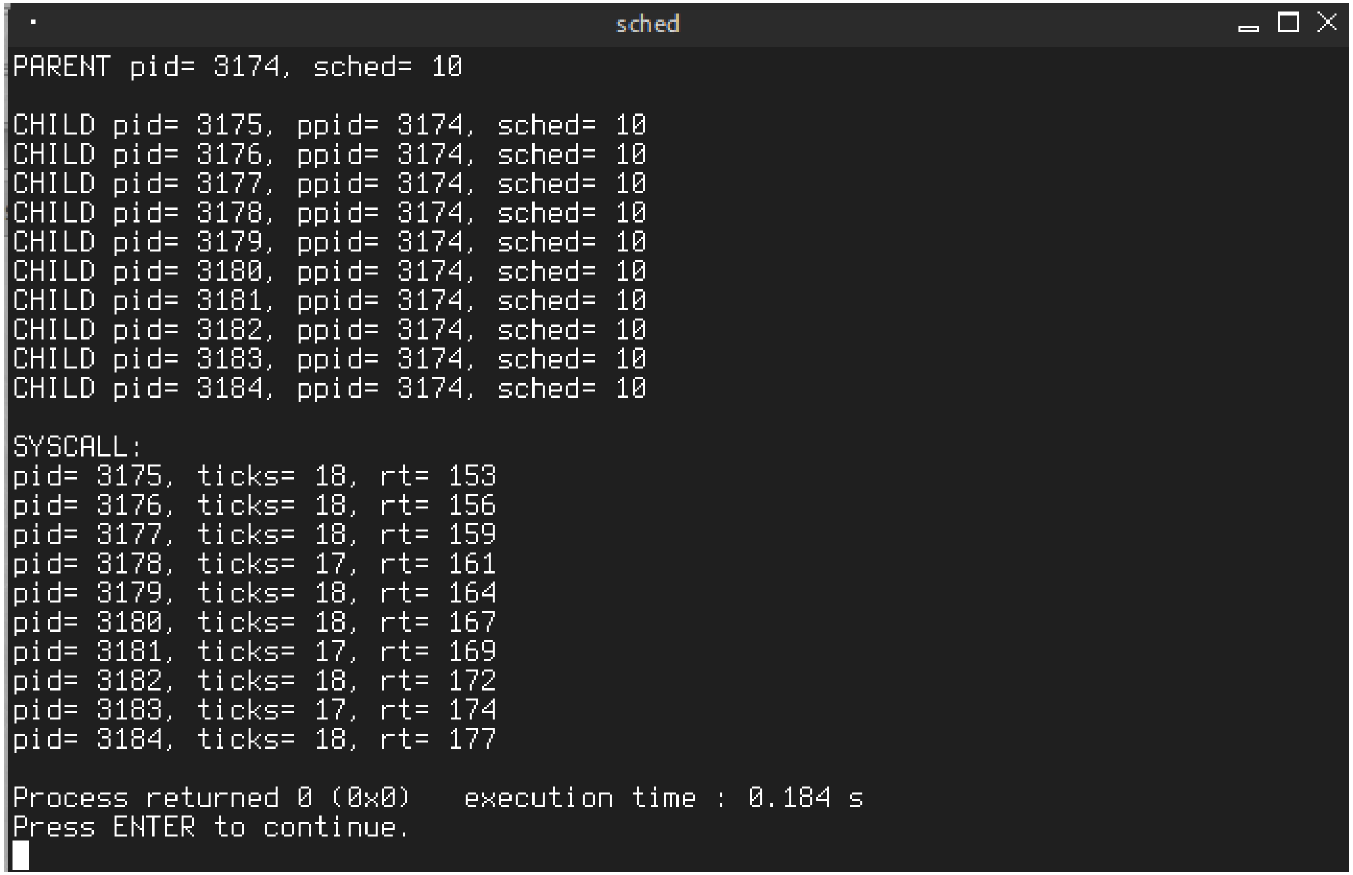
```c
    for(i = 0; i < KIDS; ++i)
    {
        p = fork();

        if(p > 0)
            continue;
        else if(p == -1)
        {
            perror("fork() failed!");
            continue;
        }
        else
        {
            for(j = 0; j < WORKLOAD; ++j)
            {
                k = rand() * rand();
                k = rand() / rand();
                k = rand() - rand();
                k = rand() + rand();
            }

            printf("CHILD pid= %d, ppid= %d, sched= %d\n", getpid(), getppid(), sched_getscheduler(0));
            exit(0);
        }

    }

```

Korisnički program - part 3



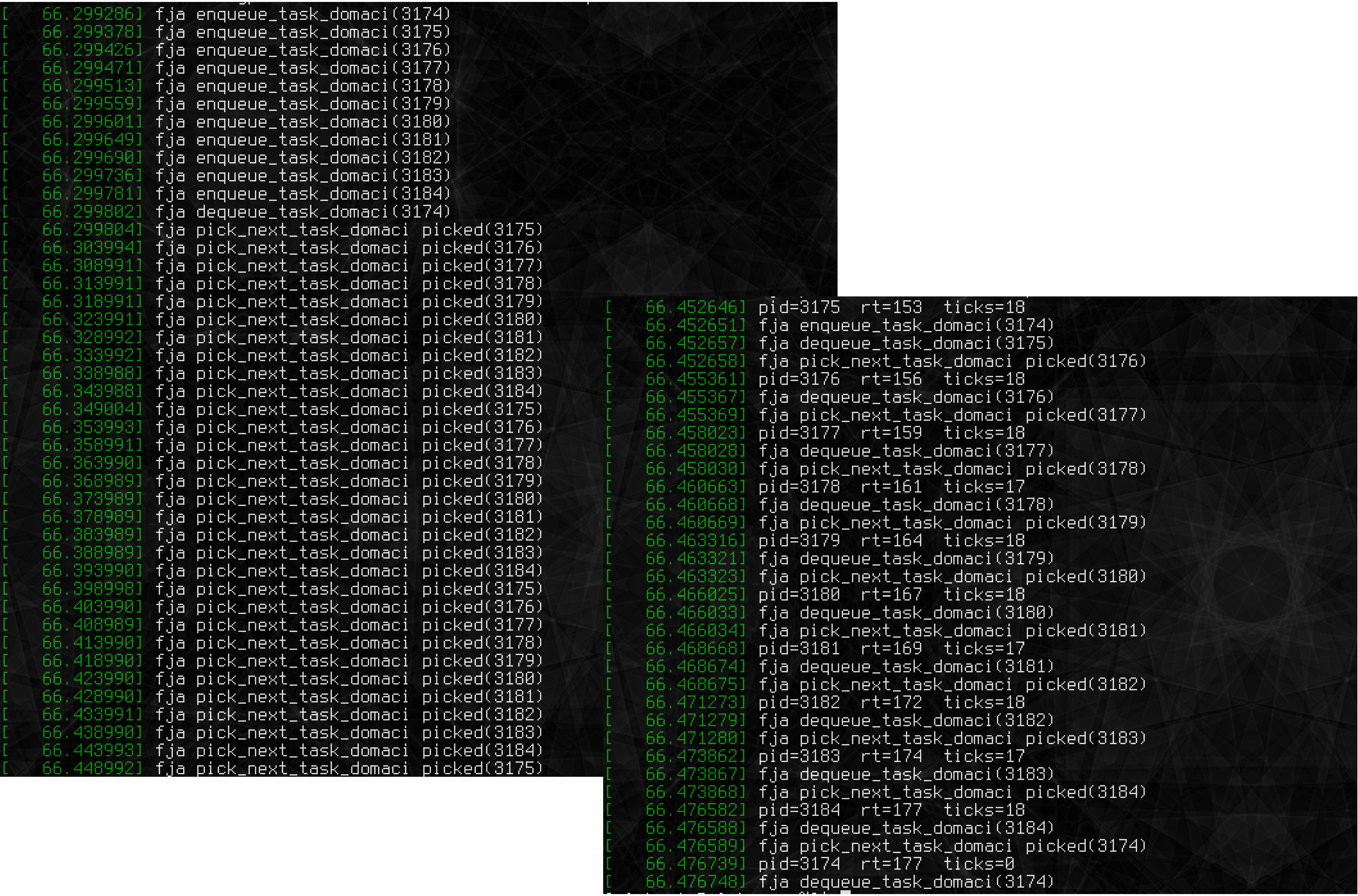
```c
    for(i = 0; i < KIDS; ++i)
        wait();

    if (syscall(400, niz) != 0)
    {
        perror("sys_domaci() failed!");
        exit(EXIT_FAILURE);
    }

    printf("\nSYSCALL:\n");
    for(i = 0; i < PROCESS_BUFF_LEN; ++i)
    {
        if(!niz[i])
            break;

        printf("pid= %d, ticks= %ld, rt= %ld\n", niz[i], niz[i + PROCESS_BUFF_LEN], niz[i + 2*PROCESS_BUFF_LEN]);
    }

    exit(0);
}
```

Korisnički program
Rezultat izvršenja



```
PARENT pid= 3174, sched= 10

CHILD pid= 3175, ppid= 3174, sched= 10
CHILD pid= 3176, ppid= 3174, sched= 10
CHILD pid= 3177, ppid= 3174, sched= 10
CHILD pid= 3178, ppid= 3174, sched= 10
CHILD pid= 3179, ppid= 3174, sched= 10
CHILD pid= 3180, ppid= 3174, sched= 10
CHILD pid= 3181, ppid= 3174, sched= 10
CHILD pid= 3182, ppid= 3174, sched= 10
CHILD pid= 3183, ppid= 3174, sched= 10
CHILD pid= 3184, ppid= 3174, sched= 10

SYSCALL:
pid= 3175, ticks= 18, rt= 153
pid= 3176, ticks= 18, rt= 156
pid= 3177, ticks= 18, rt= 159
pid= 3178, ticks= 17, rt= 161
pid= 3179, ticks= 18, rt= 164
pid= 3180, ticks= 18, rt= 167
pid= 3181, ticks= 17, rt= 169
pid= 3182, ticks= 18, rt= 172
pid= 3183, ticks= 17, rt= 174
pid= 3184, ticks= 18, rt= 177

Process returned 0 (0x0)   execution time : 0.184 s
Press ENTER to continue.
```

$ dmesg



```
[   66.299286] fja enqueue_task_domaci(3174)
[   66.299378] fja enqueue_task_domaci(3175)
[   66.299426] fja enqueue_task_domaci(3176)
[   66.299471] fja enqueue_task_domaci(3177)
[   66.299513] fja enqueue_task_domaci(3178)
[   66.299559] fja enqueue_task_domaci(3179)
[   66.299601] fja enqueue_task_domaci(3180)
[   66.299649] fja enqueue_task_domaci(3181)
[   66.299690] fja enqueue_task_domaci(3182)
[   66.299736] fja enqueue_task_domaci(3183)
[   66.299781] fja enqueue_task_domaci(3184)
[   66.299802] fja dequeue_task_domaci(3174)
[   66.299804] fja pick_next_task_domaci  picked(3175)
[   66.303994] fja pick_next_task_domaci  picked(3176)
[   66.308991] fja pick_next_task_domaci  picked(3177)
[   66.313991] fja pick_next_task_domaci  picked(3178)
[   66.318991] fja pick_next_task_domaci  picked(3179)
[   66.323991] fja pick_next_task_domaci  picked(3180)
[   66.328992] fja pick_next_task_domaci  picked(3181)
[   66.333992] fja pick_next_task_domaci  picked(3182)
[   66.338988] fja pick_next_task_domaci  picked(3183)
[   66.343988] fja pick_next_task_domaci  picked(3184)
[   66.349004] fja pick_next_task_domaci  picked(3175)
[   66.353993] fja pick_next_task_domaci  picked(3176)
[   66.358991] fja pick_next_task_domaci  picked(3177)
[   66.363990] fja pick_next_task_domaci  picked(3178)
[   66.368989] fja pick_next_task_domaci  picked(3179)
[   66.373989] fja pick_next_task_domaci  picked(3180)
[   66.378989] fja pick_next_task_domaci  picked(3181)
[   66.383989] fja pick_next_task_domaci  picked(3182)
[   66.388989] fja pick_next_task_domaci  picked(3183)
[   66.393990] fja pick_next_task_domaci  picked(3184)
[   66.398998] fja pick_next_task_domaci  picked(3175)
[   66.403990] fja pick_next_task_domaci  picked(3176)
[   66.408989] fja pick_next_task_domaci  picked(3177)
[   66.413990] fja pick_next_task_domaci  picked(3178)
[   66.418990] fja pick_next_task_domaci  picked(3179)
[   66.423990] fja pick_next_task_domaci  picked(3180)
[   66.428990] fja pick_next_task_domaci  picked(3181)
[   66.433991] fja pick_next_task_domaci  picked(3182)
[   66.438990] fja pick_next_task_domaci  picked(3183)
[   66.443993] fja pick_next_task_domaci  picked(3184)
[   66.448992] fja pick_next_task_domaci  picked(3175)
```

```
[   66.452646] pid=3175  rt=153  ticks=18
[   66.452651] fja enqueue_task_domaci(3174)
[   66.452657] fja dequeue_task_domaci(3175)
[   66.452658] fja pick_next_task_domaci  picked(3176)
[   66.455361] pid=3176  rt=156  ticks=18
[   66.455367] fja dequeue_task_domaci(3176)
[   66.455369] fja pick_next_task_domaci  picked(3177)
[   66.458023] pid=3177  rt=159  ticks=18
[   66.458028] fja dequeue_task_domaci(3177)
[   66.458030] fja pick_next_task_domaci  picked(3178)
[   66.460663] pid=3178  rt=161  ticks=17
[   66.460668] fja dequeue_task_domaci(3178)
[   66.460669] fja pick_next_task_domaci  picked(3179)
[   66.463316] pid=3179  rt=164  ticks=18
[   66.463321] fja dequeue_task_domaci(3179)
[   66.463323] fja pick_next_task_domaci  picked(3180)
[   66.466025] pid=3180  rt=167  ticks=18
[   66.466033] fja dequeue_task_domaci(3180)
[   66.466034] fja pick_next_task_domaci  picked(3181)
[   66.468668] pid=3181  rt=169  ticks=17
[   66.468674] fja dequeue_task_domaci(3181)
[   66.468675] fja pick_next_task_domaci  picked(3182)
[   66.471273] pid=3182  rt=172  ticks=18
[   66.471279] fja dequeue_task_domaci(3182)
[   66.471280] fja pick_next_task_domaci  picked(3183)
[   66.473862] pid=3183  rt=174  ticks=17
[   66.473867] fja dequeue_task_domaci(3183)
[   66.473868] fja pick_next_task_domaci  picked(3184)
[   66.476582] pid=3184  rt=177  ticks=18
[   66.476588] fja dequeue_task_domaci(3184)
[   66.476589] fja pick_next_task_domaci  picked(3174)
[   66.476739] pid=3174  rt=177  ticks=0
[   66.476748] fja dequeue_task_domaci(3174)
```