include/minix/callnr.h

```
#define SETPRI        70
```

Dodano wywołanie SETPRI.

src/mm/table.c

```
do_setpri,          /* 70 = setpri   */
```

Dodano adres wywołania.

src/mm/proto.h

```
_PROTOTYPE( void do_setpri, (void));
```

Dodano prototyp funkcji.

src/mm/main.c

```
PUBLIC void do_setpri(void) {
  message msg;
  msg = mm_in;
  _taskcall(SYSTASK, SYS_SETPRI, &msg);
}
```

Dodano funkcję do_setpri.

src/proc.h

```
  int curr_pri;
  int base_pri;
  int proc_type;
};

int MAX_PRI;
int MIN_PRI;

#define TYPE1 1
#define TYPE2 2
#define TYPE3 3
```

Dodano do struktury proc 3 wartości przechowujące informacje o procesie.
curr_pri – bieżący priorytet
base_pri – bazowy priorytet
proc_type – typ procesu, definiuje typ w jakim jest kolejkowany

Ponadto zdefiniowane stałe MIN i MAX_PRI określające granice dla procesów typu 2.

Zdefiniowano też stałe służące do nadawania typów procesów.

TYPE1 – roun drobin
TYPE2 – starzenie
TYPE3 – sjf

src/kernel/main.c

```
MAX_PRI = 1000;
MIN_PRI = 100;
/* Clear the process ta
```

Zainicjowano stałe globalne.

```
  rp->base_pri = MAX_PRI+1;
  rp->curr_pri = MAX_PRI+1; /* domyslnie rr*/
  rp->proc_type = TYPE1;
```

I ustawiono domyślne wartości procesów. Domyślnie procesy użytkownika są szeregowane algorytmem round robin.

```
proc[NR_TASKS+INIT_PROC_NR].base_pri = MAX_PRI+1;
proc[NR_TASKS+INIT_PROC_NR].curr_pri = MAX_PRI+1;
proc[NR_TASKS+INIT_PROC_NR].proc_type = TYPE1;
```

src/kernel/system.c

```
FORWARD _PROTOTYPE( int do_setpri, (message *m_ptr));
/*-----------------------------------------------------------
```

Dodano prototyp wywołania setpri.

```
case SYS_SETPRI: r= do_setpri(&m); break;
```

Obsługę jego funkcji.

Oraz funkcję.

```
PRIVATE int do_setpri(m_ptr)
message *m_ptr;
{
    struct proc* temp;

        for(temp = BEG_USER_ADDR; temp < END_PROC_ADDR; ++temp){
                if(temp->p_pid == m_ptr->m1_i3) break;
        }
    if(m_ptr->m1_i1 == 1){
        temp->curr_pri = m_ptr->m1_i2;
        temp->base_pri = m_ptr->m1_i2;
    }
    if(m_ptr->m1_i1 == 2){
        MAX_PRI = m_ptr->m1_i2;
    }
    if(m_ptr->m1_i1 == 3){
        MIN_PRI = m_ptr->m1_i2;
    }
    if(m_ptr->m1_i1 == 4){
        temp->proc_type = m_ptr->m1_i2;
    }
    return OK;
}
```

```
rpc->base_pri = rpp->base_pri;
rpc->curr_pri = rpp->base_pri;
rpc->proc_type = rpp->proc_type;
```

W funkcji fork dodano kopiowanie dodanych pól z procesu parent do procesu child.

src/kernel/proc.c Dodano obsługę zadanego algorytmu szeregującego.