Zadanie

Zrealizować algorytm szeregowania dzielący procesy użytkownika na trzy grupy: procesy interaktywne, procesy obliczeniowe oraz procesy wejścia/wyjścia.

Grupa procesów iteraktywnych otrzymuje 5 razy więcej czasu niż grupa procesów wejścia/wyjścia.

Grupa procesów obliczeniowych otrzymuje 2 razy więcej czasu niż grupa procesów wejścia/wyjścia.

Rozróżnianie grup

src/kernel/proc.h

Aby móc rozróżnić grupy procesów trzeba:

- zdefiniować możliwe grupy, jakie można przydzielić procesom USER,
- do struktury `proc` dodać pole `group` które będzie przechowywało numer grupy danego procesu oraz pole `counter` które będzie przechowywało liczbę kwantów czasu pozostałych do wykonania danego procesu

Algorytm szeregowania

Aby zachować stosunek przyznanego czasu pomiędzy grupami procesów trzeba stworzyć wywołanie systemowe `SETSCHEDULE`, które w zależności od liczby procesów każdej grupy zapewni poprawne ustawienie liczby cykli (`counter`) dla każdego procesu.

```
int do_setschedule(){
    int i = 0, num_of_int = 0, num_of_cal = 0, num_of_io = 0;
    int sum_time_int = 0, sum_time_cal = 0, sum_time_io = 0;

    // Liczenie liczby procesów każdej grupy
    for (i;i<NR_PROCS; i++){
        if (proc[i].group == 'interaction')
            num_of_int++;
        if (proc[i].group == 'calculation')
            num_of_cal++;
        if (proc[i].group == 'IO')
            num_of_io++;
    }

    // Liczenie czasu który musi być przeznany każdej grupie
    if ((num_of_int == 5*num_of_io && num_of_cal == 2*num_of_io)
        || (num_of_int < 5*num_of_io && num_of_cal < 2*num_of_io)){
        sum_time_io = num_of_io;
        sum_time_int = sum_time_io*5;
        sum_time_cal = sum_time_io*2;
    }
    else if (num_of_int > 5*num_of_io && num_of_cal <= 2*num_of_io){
        sum_time_int = closest_devided_by(num_of_int, 5);
        sum_time_io = sum_time_int / 5;
        sum_time_cal = 2*sum_time_io;
    }
```

```
    else if (num_of_cal > 2*num_of_io && (num_of_cal > num_of_int || num_of_cal
<= num_of_int)){
        sum_time_cal = closest_devided_by(num_of_cal, 2);
        sum_time_io = sum_time_cal / 2;
        sum_time_int = sum_time_io * 5;
    }

    sum_time_io -= num_of_io;
    sum_time_int -= num_of_int;
    sum_time_cal -= num_of_cal;

    // Ustawienie wartości 'counter' dla każdego procesu grupy interaktywnej
    while(sum_time_int != 0){
        for (i=0; i<NR_PROCS; i++){
            if (proc[i].group == "interaction"){
                proc[i].counter++;
                sum_time_int--;
                if (sum_time_int == 0){
                    break;
                }
            }
        }
    }
    // Ustawienie wartości 'counter' dla każdego procesu grupy obliczeniowej
    while(sum_time_cal != 0){
        for (i=0; i<NR_PROCS; i++){
            if (proc[i].group == "calculation"){
                proc[i].counter++;
                sum_time_cal--;
                if (sum_time_cal == 0){
                    break;
                }
            }
        }
    }
    // Ustawienie wartości 'counter' dla każdego procesu grupy wejścia/wyjścia
    while(sum_time_io != 0){
        for (i=0; i<NR_PROCS; i++){
            if (proc[i].group == "io"){
                proc[i].counter++;
                sum_time_io--;
                if (sum_time_io == 0){
                    break;
                }
            }
        }
    }

}
```

Przykłady działania:

- Zostało uruchomiono 5 procesów grupy interaktywnej, 5 procesów grupy obliczeniowej i 4 procesy grupy we/wy. Wtedy grupa we/wy otrzymuje 4 kwanty czasu procesora, grupa obliczeniowa 4*2=8 kwantów czasu, grupa interaktywna 4*5=20 kwantów.
- Zostało uruchomiono 5 procesów grupy interaktywnej, 5 procesów grupy obliczeniowej i 1 procesy grupy we/wy. W tej sytuacji nie możemu już przydzielić procesom grupy we/wy 1

kwant czasu i odpowiednio pozostałym, bo w takim razie grupa obliczeniowa dostałaby 4 kwanty czasu, co jest niewystarczająco przy pięciu procesach. Dlatego dla liczby procesów obliczeniowych szukamy liczby całkowitej większej niż 5, która jest podzielna przez 2, żeby dostać całkowitą liczę kwantów przydzielonych procesowi grupy we/wy. Więc grupa obliczeniowa dostaje 6 kwantów czasu, grupa we/wy - 6/2=3 kwanty, grupa interaktywna - 3*5=15 kwantów.

src/kernel/proc.c

Zmodyfikować funkcje `sched()` tak aby sprawdzała ona wartość pola `counter` procesu. Jeśli nie jest one równe zeru, wtedy ten proces otrzymuje kolejny kwant czasu, a wartość pola `counter` jest dekrementowana o jeden. Jeśli `counter` = 0, wtędy czas procesora otrzymuje następny proces.

```
PRIVATE void sched()
{
    if (++rdy_head[USER_Q]->counter > 0){
        pick_proc();
    }
    rdy_tail[USER_Q]->p_nextready = rdy_head[USER_Q];
    rdy_tail[USER_Q] = rdy_head[USER_Q];
    rdy_head[USER_Q] = rdy_head[USER_Q]->p_nextready;
    rdy_tail[USER_Q]->p_nextready = NIL_PROC;
}
```

Zmiana grup procesów

src/kernel/system.c

Wszystkie procesy podczas inicjalizacji zostają przydzieleni do grupy bazowej, która ma jeden kwant czasu na wykonanie oraz mają ustawioną wartość licznika (`counter`) na 1.

src/mm/

Aby móc przenosić procesy do różnych grup trzeba stworzyć wywołanie systemowe `SETPROCGROUP`. W tym celu należy:

- Do pliku **include/minix/callnr.h** oraz **include/minix/com.h** dodać odpowiednio numer syscallu i taskcallu.
- W pliku **src/kernel/system.c** umieścić procedurę obsługi `SETPROCGROUP`.
- W pliku **src/mm/main.c** umieścić procedurę obsługi do_setprocgroup oraz prototyp tej funkcji umieścić w pliku **src/mm/proto.h**.
- W pliku **/src/mm/table.c** w tablicy call_vec w odpowiednim miejscu wstawić nazwę funkcji do_setprocgroup, zaś w pliku **src/fs/table.c** w tym samym miejscu umieścić adres pusty funkcji, no_sys.

Testowanie

Aby przetestować algorytm szeregowania należy stworzyć kilka programów, które będą tworzyły procesy określonej grupy a następnie ich uruchomić:

```
./int1&;./int2&;...;./intN&; ./cal1&;./cal2&;...;./calN&;
./io1&;./io2&;...;./ioN&
```

Następnie, wciskając klawisz F1 śledzić wykonanie tych procesów. Po jakimś czasie stosunek sum czasów wykonania grupy procesów interaktywnych do grupy procesów obliczeniowych do grupy procesów wejścia/wyjścia będzie 5:2:1