Ćwiczenie 2.

Szeregowanie procesów

# 1. Cel ćwiczenia

Celem ćwiczenia jest zaprojektowanie mechanizmu szeregowania w systemie MINIX. W trakcie ćwiczenia należy zamienić standardową procedurę szeregującązgodnie z algorytmem szeregowania o cechach wskazanych przez prowadzącego.

## 2. Szeregowanie w systemie MINIX

W systemie MINIX procesy s± podzielone na trzy klasy:

* zadania systemowe (TASK)
* procesy serwerowe (SERVER)
* procesy użytkowe (USER)

Procesy TASK maja bezwzględny priorytet nad procesami SERVER, a te z kolei nad procesami USER. Procesy systemowe TASK i SERVER s± realizowane w reżimie pobłażającym, to znaczy proces realizuje się tak długo, aż sam zrzecze się procesora. Proces USER po wyczerpaniu kwantu czasu jest przenoszony na koniec kolejki procesów gotowych. Procedura szeregująca wybiera do realizacji zawsze proces będący głową kolejki procesów gotowych.

# 3. Przykładowy algorytm szeregowania

Poniżej przedstawiono przykładowy algorytm szeregowania. Proszę pamiętać, że jest to tylko przykład, a właściwe zadanie do zrealizowania w ramach laboratorium znajduje się w punkcie 6.

W przykładowym algorytmie szeregowania obowiązują następujące zasady:

* proces ma dwa atrybuty: priorytet bazowy BASE\_PRI i priorytet aktualny ACT\_PRI.
* dwie zmienne systemowe MAX\_AGE i MIN\_PRI, MAX\_AGE>MIN\_PRI, dzielą szeregowane procesy na trzy kategorie:
  + priorytet bazowy > MAX\_AGE - proces realizowany w reżimie pobłażania (oczywiście z wyjątkiem procesów klas TASK i SERVER), priorytet bieżący zawsze równy bazowemu,
  + priorytet bazowy > MIN\_PRI, <= MAX\_AGE - proces realizowany w reżimie starzenia: proces któremu jest zabierany procesor otrzymuje priorytet bieżący równy bazowemu, wszystkie inne z tej grupy zwiększają priorytet bieżący o 1 (maksymalnie do MAX\_AGE); proces któremu jest zabierany procesor jest wstawiany do kolejki za innymi procesami o tym samym priorytecie bieżącym,
  + priorytet bazowy < MIN\_PRI - proces realizowany w reżimie priorytetów statycznych z podziałem czasu: proces któremu jest zabierany procesor jest wstawiany do kolejki za innymi procesami o tym samym priorytecie bieżącym; priorytet bieżący zawsze równy bazowemu.

# 4. Cechy funkcjonalne przykładowego algorytmu w systemie MINIX:

* realizacja szeregowania zgodnie z przedstawionym algorytmem,
* wartości początkowe zmiennych: MAX\_AGE=1000, MIN\_PRI=100,
* każdy nowo tworzony proces otrzymuje priorytet bazowy i bieżący równe MIN\_PRI,
* system udostępnia nowe wywołanie - ustaw parametr szeregowania:  
  void set\_PRI (int nr; unsigned VALUE)

nr = 1 nadaj bieżącemu procesowi priorytet VALUE

nr = 2 ustaw zmienna MAX\_AGE na VALUE

nr = 3 ustaw zmienna MIN\_PRI na VALUE

# 5. Wskazówki implementacyjne przy implementacji algorytmu przykładowego

1. plik kernel/proc.h:

* w strukturze 'proc' umieścić priorytet bazowy i bieżący procesu,
* zadeklarować zmienne systemowe MAX\_AGE i MIN\_PRI.

1. plik kernel/proc.c:

* w funkcji 'sched()', która jest procedurę szeregującą, zamiast przenoszenia procesu bieżącego na koniec kolejki procesów gotowych zrealizować zasady algorytmu przykładowego,
* w funkcji 'ready(rp)', która budzi proces, zadbać o umieszczenie go w odpowiednim miejscu struktur algorytmu (jeżeli jest to proces klasy USER).

1. plik kernel/system.c:

* w funkcji 'do\_fork(m\_ptr)', która tworzy nowy proces, w części dotyczącej procesów klasy USER zadbać o odpowiednie zainicjowanie priorytetu bazowego i bieżącego procesu.

1. plik kernel/main.c:

* w funkcji 'main()', na pocz±tku zadbać o właściwe zainicjowanie zmiennych systemowych MAX\_AGE i MIN\_PRI, w pętli powołującej deskryptory procesów (for (rp=BEG...)) zadbać o właściwe wypełnienie priorytetów, w części końcowej powołującej proces INIT zadbać również o odpowiednie wartości priorytetów.

1. dodać do systemu wywołanie systemowe zgodnie ze składnię podaną w punkcie 4, z tym, że odwołanie ma nastąpić do samego mikrojądra za pośrednictwem jednego z modułów MM/FS. Dlatego też funkcja realizująca dodane wywołanie systemowe (np. o nazwie do\_setpri()) w jednym z modułów MM/FS powinna mieć poniższa postać:

PUBLIC void do\_setpri()

{

message m;

m = mm\_in;

\_taskcall(SYSTASK, SYS\_SETPRI, &m);

}

gdzie SYS\_SETPRI jest zdefiniowana przez nas stała w pliku/usr/include/minix/com.h.   
Ponadto, do pliku /usr/src/kernel/system.c należy dodać obsługę funkcji o numerze SYS\_SETPRI i zdefiniować sam± funkcję obsługującą to wywołanie. W obsłudze można wykorzystać makro proc\_addr( id ) zwracające wskazanie na strukturę proc zadanego procesu.

# 6. Zadanie do zrealizowania

Prowadzący poleci studentom zrealizować algorytm szeregowania o zadanych własnościach funkcjonalnych. Przykład: zrealizować algorytm szeregowania który zapewni właściwe traktowanie procesów trzech kategorii: interaktywnych (np. gra w statki), obliczeniowych (obliczenia numeryczne) oraz masywnego wejścia/wyjścia realizowanego w tle (np. proces składowania dysku na taśmie).