Systemy Operacyjne - laboratorium

Ćwiczenie 2 - Szeregowanie procesów

Paweł Bęza

1. Treść zadania

   Celem ćwiczenia jest zaprojektowanie mechanizmu szeregowania w systemie MINIX. W trakcie ćwiczenia należy zamienić standardową procedurę szeregującą zgodnie z algorytmem szeregowania o cechach wskazanych przez prowadzącego. Zrealizować algorytm szeregowania o zadanych własnościach funkcjonalnych. Należy dodać do procesów użytkowych trzy nowe typy: a) wsadowy - w ramach tej grupy programy są szeregowane pobłażliwie, tj. proces otrzymuje procesor aż do swojego ukończenia, b) zwykły - należy wykorzystać algorytm starzenia opisany w instrukcji oraz c) interaktywny - obowiązuje algorytm round-robin, czyli procesy wykonywane są kolejno po kwancie (proces wywłaszczony umieszczany jest na końcu kolejki; do wykonania brany jest proces pierwszy w kolejce)

2. W celu zaimplementowania rozwiązania spełniające warunki polecenia zmienię poniższe pliki:

- **kernel/proc.h**
  - w strukturze proc dodam pole **p_type**, które będzie mówiło o typie procesu (wsadowy, starzenia lub round-robin). Oprócz tego pola dodam również pola **BASE_PRI** oraz **ACT_PRI**, które będą używane w algorytmie starzenia opisanym w skrypcie
  - zadeklaruję zmienne systemowe **MAX_AGE** i **MIN_PRI**, które również będą używane w algorytmie starzenia
- **kernel/proc.c**
  - w funkcji **pick_proc()** zmiana wartości **bill_ptr** w zależności od typu procesu
  - w funkcji **sched()** zamiast przenoszenie procesu bieżącego na koniec kolejki procesów gotowych realizuje zasady algorytmu z zadania
  - w funkcji **ready(rp),** umieszczam wybudzony proces w odpowiednim miejscu kolejki tak, aby kolejka była podzielona grupami na typ interaktywny, normalny oraz wsadowy
- **kernel/system.c**
  - zainicjalizowanie typu procesu wartością domyślną reprezentującą algorytm starzenia
  - zainicjalizowanie pól **BASE_PRI** oraz **ACT_PRI** wartością **MIN_PRI**
- **kernel/main.c**
  - zainicjalizowanie zmiennych systemowych **MAX_AGE** i **MIN_PRI**
  - w pętli powołującej deskryptory procesów ustawiam priorytety **BASE_PRI** oraz **ACT_PRI** na **MIN_PRI**
  - w części końcowej powołującej proces **INIT** ustawiam priorytety **BASE_PRI** i **ACT_PRI** na **MIN_PRI** oraz ustawiam typ procesu **INIT** na typ interaktywny
- **wywołanie systemowe**

- dodanie do serwera MM wywołania systemowego:
  - w */usr/include/minix/callnr.h* dodaje nowy identyfikator procesu **SETPRI** i zwiększam o jeden stałą **N_CALLS**
  - w */usr/src/mm/table.c* dodaje w tablicy *call_vec* w odpowiednim miejscu wstawiam adres funkcji **do_getprocnr**, zaś w pliku */usr/src/fs/table.c* w tym samym miejscu umieszczam adres pusty funkcji **no_sys**
  - w */usr/src/mm/proto.h* dodaje prototyp funkcji **do_setpri**
  - w */usr/src/mm/main.c* umieszczam definicje funkcji **do_setpri**
- dodanie wywołania do mikrojądra
  - w */usr/include/minix/com.h* dodaje nowy identyfikator procesu **SYS_SETPRI**
  - w /usr/src/kernel/system.c definiujemy prototyp **do_setpri,** w funkcji **sys_task** dodajemy wywołanie oraz definiujemy wywołanie **do_setpri**