Wojciech Sitek

SOI LAB - Koncepcja zadania 2

# Treść i założenia wstępne zadania

1. Przygotować algorytm szeregowania procesów w systemie Minix z następującymi założeniami:
   1. w systemie będą występować trzy kategorie procesów: minPRI, midPRI i highPRI
   2. do ustawiania kategorii procesu należy przygotować wywołanie systemowe set\_PRI
   3. procesy będą realizowane w statycznym reżimie czasu, a więc będą mieć przydzielane kwanty czasu
   4. procesy highPRI powinny mieć możliwość wykonywania przez 3 kolejne kwanty czasu
   5. procesy midPRI powinny mieć możliwość wykonywania przez 2 kolejne kwanty czasu
   6. procesy minPRI po upłynięciu kwantu czasu od razu lądują na końcu kolejki
2. Test działania szeregowania:
   1. Przygotować program/skrypt, który będzie powoływał zadaną liczbę procesów o zadanych priorytetach.
   2. Na koniec testu powinna być wypisywana informacja o czasie pracy każdego z procesów (najlepiej procent czasu w jakim pracował proces z całego czasu testu).

# Założenia własne

* Kategorie procesów będą nadawane tylko procesom typu USER

# Propozycja realizacji założeń

* W strukturze procesu (/usr/src/kernel/proc.h) umieszczenie zmiennej definiującej kategorię procesu (PCAT\_NONE=0, PCAT\_MIN=1, PCAT\_MID=2, PCAT\_HIGH=3)
* W pliku zarządzającym procesami (/usr/src/kernel/proc.c) deklaracja zmiennej statycznej *int time\_left* (na zewnątrz funkcji), pokazującej, ile kwantów czasu należy jeszcze przeznaczyć dla obecnie wykonywanego procesu.
* W funkcji *sched()*, szeregującej procesy (/usr/kernel/proc.c) implementacja szeregowania tak, aby:
  + Dla procesu, który ma właśnie otrzymać przydzielony czas procesora, gdy jest to proces ze zdefiniowaną wartością kategorii, przypisać do *time\_left* zdekrementowaną o 1 łączną ilość kwantów czasu (dla PCAT\_MIN=>0, dla PCAT\_MID=>1, dla PCAT\_HIGH=>2)
  + Dla działającego procesu o niezerowej wartości *time\_left* (PCAT\_MID lub PCAT\_HIGH) dekrementacja o 1 wartości *time\_left* i pozostawienie tego samego procesu.
  + Dla działającego procesu o zerowej wartości *time\_left* (PCAT\_MIN lub nieustawiona kategoria, czyli PCAT\_NONE, lub PCAT\_MID/PCAT\_HIGH po upłynięciu 2/3 kwantów czasu) wyrzucenie procesu na koniec kolejki i pobranie procesu pierwszego z kolejki
* Implementacja nie potrzebuje zmiany funkcji *ready(rp)*, ponieważ działa ona na tej samej kolejce procesów, jedynie przedłużając ich czas przydzielania.
* Ustawienie priorytetu nowego procesu na wartość podaną w strukturze message przy tworzeniu nowego procesu w funkcji do\_fork(m\_ptr) (/usr/src/kernel/system.c)
* Utworzenie wywołania systemowego ustawiającego jedną z kategorii procesu:
  + Napisanie funkcji do\_setpri() (/usr/src/mm/main.c) jako przekazania zadania do jądra przez stałą SYS\_SETPRI (/usr/include/minix/com.h)
  + Dostosowanie nagłówków do nowego wywołania systemowego (callnr.h, table.c, itp.)
  + Napisanie funkcji o numerze SYS\_SETPRI obsługującej ustawianie priorytetu danego procesu (/usr/src/kernel/system.c)
* Rekompilacja jądra
* Napisanie programu testowego w C o następujących funkcjach:
  + Pobranie argumentów w postaci: [liczba procesów MIN\_PRI] [liczba procesów MID\_PRI] [liczba procesów HIGH\_PRI] [ew. czas testu w s]
  + Utworzenie zadanej liczby procesów o zadanych priorytetach
  + Czekanie zadaną ilość czasu na pracę procesora
  + Wypisanie informacji o procencie czasu, w jakim pracował każdy z procesów
* Testowanie implementacji