Wojciech Celej

Nr albumu: 271248

KONCEPT DO LABORATORIUM NR 2 – SZEREGOWANIE PROCESÓW

1. Zadanie

Podział procesów na 3 grupy: A, B i C. Grupy A i B oraz C.
Grupa A - dostaje tym więcej czasu procesora im wyższy ma priorytet (przyjmiemy zakres 1-10)
Grupa B - algorytm starzenia
Grupa C - wywłaszczanie, jakie się pojawia taki proces - ma pierwszeństwo przed A i B

2. Rozwiązanie

- Będziemy potrzebować wywołania systemowego do mikrojądra (które będzie obsługiwane pośrednio przez moduł MM). Dodamy funkcję do_setpriority, która będzie obsługiwać wywołanie systemowe ustawiające priorytet bazowy i aktualny.
  Funkcja ta umożliwi również ustawienie wartości systemowych MAX_AGE i MIN_PRI (jeżeli priorytet bazowy procesu > od MAX_AGE, to proces będzie zakwalifikowany do grupy C, jeżeli < MIN_PRI, to do grupy A). Napiszę wywołanie korzystające w tej funkcji (napisanie procedur je obsługujących, dodanie odpowiednich wpisów w tablicach wykonam tak jak na zadaniu na LAB 1.)

- W pliku src/kernel/proc.h dodam pola na priorytet bazowy oraz aktualny

- W pliku src/kernel/main.c inicjuję zmienne MAX_AGE i MIN_PRI, w pętli powołującej deskryptory procesów (for (rp=BEG...)) domyślnie dodajemy procesy do grupy A

- W pliku src/kernel/system.c w funkcji 'do_fork(m_ptr)', która tworzy nowy proces, w części dotyczącej procesów klasy USER przyjmujemy, że proces potomny przyjmie wartości priorytetów od rodzica; dla procesu INIT musimy nadać wartości domyślne (bo nie powstaje przez FORK)

- Kwintesencja - plik kernel/proc.c

Funkcja sched () to planista systemu Minix. Pomysł – taka modyfikacja tej funkcji oraz tej pick_proc () aby zamiast algorytmu karuzelowego, czas był przydzialny proporcjonalnie do priorytetu. Zatem przy wystąpieniu przerwania zegarowego (po upływie 100ms) planista podejmie decyzję, który procesor wstawić do głowy kolejki, na podstawie priorytetu bieżącego. Potrzebna będzie zmienna, przechowująca liczbę kwantów czasu dla aktualnie wykonującego się procesu (jej wartość może być na starcie równa priorytetowi bieżącemu)
W funkcji ready () powinniśmy zapewnić, aby procesy z grupy C były wstawiane do kolejki procesów przed procesami z grup A i B.
Procesy, które są wywłaszczane, odzyskują oczywiście swój priorytety bieżący (dla procesów grupy B zmienia się on przy każdym wywołaniu funkcji pick_proc () zgodnie z algorytmem starzenia).

3. Testy

Powołanie do 4 procesów odpowiedni z grup A i B. Dla procesów o statycznych priorytetach (gr. A) – pomiar czasu wykonania. Dla procesów z grupy B – pomiar czasu wykonania + wyświetlenie co jakiś czas priorytetu bieżącego tak, by użytkownik mógł dostrzec zmianę.
Z poziomu shella uruchomienie 2 procesów gr. C – w tym czasie procesy z gr. A i B powinny przestać się wykonywać. Po zakończeniu pracy tych procesów – powinny wystartować. Procesom z grup C również mierzymy czas.