Zarządzanie procesami

Systemy Operacyjne

Kacper Modelski

06.11.2019

Wymagania

- Moduł zarządzania procesami wzorowany na systemie operacyjnym Windows
  - Windows: CreateProcess -> Linux: fork i exec
- Tworzenie nowych procesów
- Kończenie procesów
- Zmiana priorytetów procesu
- Zmiana stanu procesu

Stany procesów

- Struktura enum ProcessState
- Stany procesów są przechowywane w PCB procesu
- Możliwe stany to
  - NEW
  - READY
  - RUNNING
  - WAITING
  - ~~TERMINATED~~

PCB procesu

- Struktura przechowująca najważniejsze dane dotyczące procesu:
  - PID (Process ID – identyfikator procesu) – typu int
  - Nazwę pliku – typu string
  - Priorytet statyczny – typu int
  - Priotytet dynamiczny – typu int
  - Stan procesu – własna struktura enum ProcessState
  - Długość programu – typu int
  - Aktualną pozycje w pliku – PC (Program Counter) – typu int
  - 4 rejestry ogólnego przeznaczenia – typu int

Tworzenie procesu

- void KM_CreateProcess(string nazwa_pliku, int priorytet)
- Funkcja tworzy nowy proces, czyli:
  - Nadaje PID (Process ID – identyfikator procesu)
  - Tworzy PCB procesu i zapisuje jego referencje w odpowiednich listach – nadanie stanu NEW oraz dodanie do listy wszystkich procesów
  - Sprawdza czy na danym pliku nie jest ustawiony semafor – wywołanie odpowiedniej funkcji – jeżeli oczekuje – nadanie stanu WAITING (?)
  - Otwiera i wczytuje plik zawierający program (?)
  - Nadanie stanu READY
  - Dodanie referencji do listy procesów gotowych
- Wywoływana przez system – Interfejs, oraz assembler

Kończenie procesu

- void KM_TerminateProcess(PCB &pcb)
  - Usunięcie procesu z kolejki procesów gotowych i aktywnych
  - Zmiana stanu procesu na TERMINATED
  - Zwolnienie semafora na danym pliku – wywołanie odpowiedniej funkcji
  - Zwolnienie zasobów używanych przez proces
- Wywoływana przez system oraz po zakończeniu wykonywania procesu

Przechowywanie listy procesów

- Lista wszystkich aktywnych procesów (wszystkie procesy oprócz procesów zakończonych)
- Lista procesów gotowych
- Listy są zmiennymi typu List<&PCB> - przechowują referencje PCB procesów

Funkcje dodatkowe

- PCB KM_getPCBbyPID(PCB &pcb)
  - Zwraca PCB uzyskane po referencji PCB
  - Wywoływana przez moduł zarządzania procesorem
- List<&PCB> KM_getReadyProcessList()
  - Zwraca liste PCB procesów gotowych
  - Wywoływana przez moduł zarządzania procesorem
- List<&PCB> KM_getAllProcessList()
  - Zwraca liste PCB wszystkich procesów
  - Wywoływana przez moduł zarządzania procesorem oraz interfejs
- ~~List<PCB> KM_getWaitingProcessList()~~
  - ~~Zwraca liste PCB procesów oczekujących~~

Zmiana priorytetu procesu oraz jego stanu

- void KM_setProcessPriority(PCB &pcb, int priorytet)
  - Zmienia dynamiczny priorytet procesu w jego PCB
  - Wywoływana przez moduł zarządzania procesorem
- void KM_setProcessState(PCB &pcb, ProcessState ps)
  - Zmienia stan procesu w jego PCB
  - Wywoływana przez moduł zarządzania procesorem