Wywołania systemowe | Koncepcja wykonania zadania

Damian D'Souza

1. Wywołanie systemowe zwracające PID procesu z największą liczbą potomków w przedziale <A, B>

Aby zaimplementować wywołanie systemowe, które zwraca PID procesu posiadającego największą liczbę potomków mieszczącą się w zadanym przedziale <A, B>, można skorzystać z podejścia opartego na rekurencyjnym przeszukiwaniu tablicy procesów.

Struktura zawierająca dane o procesach, która jest zdefiniowana w pliku `usr/src/mm/mproc.h`, to tablica struktur zawierających informacje o procesach o wielkości `NR_PROCS` zdefiniowanej w pliku `usr/include/minix/config.h`. Iterując po niej można sprawdzić czy analizowana w danej pętli pozycja jest w użyciu porównując pole `mp_flags` ze stałą `IN_USE` zdefiniowaną w tym samym pliku co struktura `mproc`. Jeśli pole jest w użyciu można rekurencyjnie tak samo sprawdzać i zliczać dzieci tego procesu co można osiągnąć przekazując do każdego kolejnego wywołania funkcji zliczającej indeks rodzica i szukając dziecka porównać go z polem `mp_parent`, które zawiera indeks rodzica danego procesu.

Po znalezieniu liczby potomków spełniającej warunki zadania można ją i pid procesu przypisać do odpowiednich pól `mp_reply`, struktury `mproc`, które później można odczytać z pól struktury `message`, do której wskaźnik został przekazany do wywołania przez funkcję `_syscall`.

2. Wywołanie systemowe zwracające PID procesu z największą liczbą potomków na N poziomach

Drugie wywołanie systemowe polega na znalezieniu procesu, który posiada największą liczbę potomków w ciągu N poziomów (dzieci, wnukowie, prawnukowie, itd.). Koncepcja ta może być zrealizowana poprzez modyfikację rekurencyjnego podejścia z poprzedniego zadania.

Aby ograniczyć przeszukiwanie do określonej liczby poziomów można utworzyć zmienną wskazującą na poziom w danym wywołaniu. Po znalezieniu dziecka

zmienną tą trzeba dekrementować, a po powrocie z wywołania znowu inkrementować. Warunkiem końcowym takiej funkcji będzie wartość zmiennej równa 0.

Przykład:

Szukając potomków na 3 poziomach czyli dzieci, wnuków i prawnuków funkcja ta zachowa się następująco.

1. Początkowa wartość poziomu będzie wynosiła 3.
2. Po znalezieniu aktywnego procesu zostanie wywołana funkcja zliczająca, której zostanie przekazana początkowa wartość poziomu (3).
3. Dla każdego dziecka funkcja zliczająca dostanie wartość poziomu równą 2.
4. Dla wnuka poziom zliczania zostanie przekazany jako 1.
5. Gdy funkcja zliczająca odnajdzie dziecko prawnuka, otrzyma poziom równy 0, co spowoduje powrót do zliczania prawnuków.
6. Po zakończeniu liczenia prawnuków funkcja powróci do poprzedniego poziomu i sprawdzi dzieci kolejnego wnuka.
7. Ostatecznie funkcja zwróci sumę zliczonych potomków na określonym poziomie, a następnie porówna z wynikami dla innych procesów i zwróci wynik spełniający warunki zadania.

3. Testowanie

Aby przetestować wywołanie można w pętli tworzyć nowe procesy, aby nie wykonać wywołania wielokrotnie porównać wynik `fork()` z 0, jeśli będą równe to wstrzymać proces za pomocą `sleep()` przez kilka sekund po czym zakończyć program. Dla wyniku większego od zera powtórzyć pętlę. Taki kod utworzy n dzieci procesu, w którym działa program testujący.