Systemy operacyjne
29.10.2010
System UNIX

Liteartura dodatkowa:
Deo – Teoria grafów i jej zastosowania w technice i informatyce

---

Unix jest systemem operacyjnym z podziałem czasu (time sharing, concurrent processing) realizującym wielodostęp i wielozadaniowość. Pierwotna wersja systemu z roku 1970 pochodzi od Kena Thompsona i Dennisa Ritchiego z Bell Laboratory.

Podstawowe cechy sytemu:

- interakcyjny interfejs użytkownika stanowi program shell – może być zmieniany
- system plików jest wielopoziomowym drzewem w którym można tworzyć podkatalogi i pliki, a dostęp do danych jest bezpośredni i sekwencyjny
- pliki i urządzenia wyjścia-wejścia są traktowane jedonilicie
- system umożliwia wieloprocesorowość (parallel procesing)
- szeregowanie zadań (tasks scheduling):
  - krótkoterminowe (czasu procesorów) oparte o algorytmy priorytetowe oraz cykliczne (round-robin)
  - średnioterminowe (pamięci wirtualnej) oparte o algorytmy zmiennych obszarów z wymianą (swapping) i stronicowania (paging)
  - długoterminowe (pamięci dyskowej) na systemie plików opartym o i-węzły (inode)
- jest napisany głównie w języku c specjalnie opracowanym dla jego zaprogramowania – co daję łatwą przenośność systemu
- próbuje zapobiegać załamaniu systemu:
  - zakleszczenie (deadlock) - jest pojęciem opisującym sytuację, w której co najmniej dwie różne akcje czekają na siebie nawzajem, więc żadna nie może się zakończyć
  - starvation
  - migotanie stron – konflikt zasobowy
- pierwotny model systemu został ukierunkowany tekstowo i rozwijany w kierunku:
  - pracy z grafiką – Xwindow
  - pracy w sieciach – TCP/IP
  - pracy w czasie rzeczywistym

Za każdym razem gdy pojawia się nowa potrzeba **UNIX** potrafi się do niej przystosować wciąż pozostając **UNIX**-em.

Fragmentacja:

- Fragmentacja wewnętrzna (*internal fragmentation*) - straty pamięci powodowane nieużytkami występującymi w ostatnich blokach plików lub w końcowych stronach programu.

- Fragmentacja zewnętrzna(*external fragmentation*) - straty pamięci powodowane nieużytkami, czyli dziurami (h*oles*) występującymi na dysku lub w pamięci operacyjnej między poszczególnymi blokami informacji.

Warstwy strukturalne sytemu:

<table>
<tr><td colspan="3">Użytkownicy</td></tr>
<tr><td colspan="3">Funkcje systemowe implementują:

shelle
języki programowania (kompilatory i interpretery)
biblioteki systemowe</td></tr>
<tr><td colspan="3">Interfejs funkcji systemowych z jądrem</td></tr>
<tr><td colspan="3">Jądro implementuje:</td></tr>
<tr><td><u>Szeregowanie zadań i synchronizacja procesów:</u>

Przydział czasu procesorów
Prydział czasu przestrzeni adresowej</td><td><u>System plików:</u>

wymiana, stronicowanie
obsługa we / wy</td><td><u>Gniazdka i sygnały:</u>

protokoły
obsługa sieci</td></tr>
<tr><td colspan="3">Interfejs jądra ze sprzętem</td></tr>
<tr><td colspan="3">Sterowniki:

terminali urządzeń zewnętrznych pamięci</td></tr>
<tr><td colspan="3">Sprzęt</td></tr>
</table>

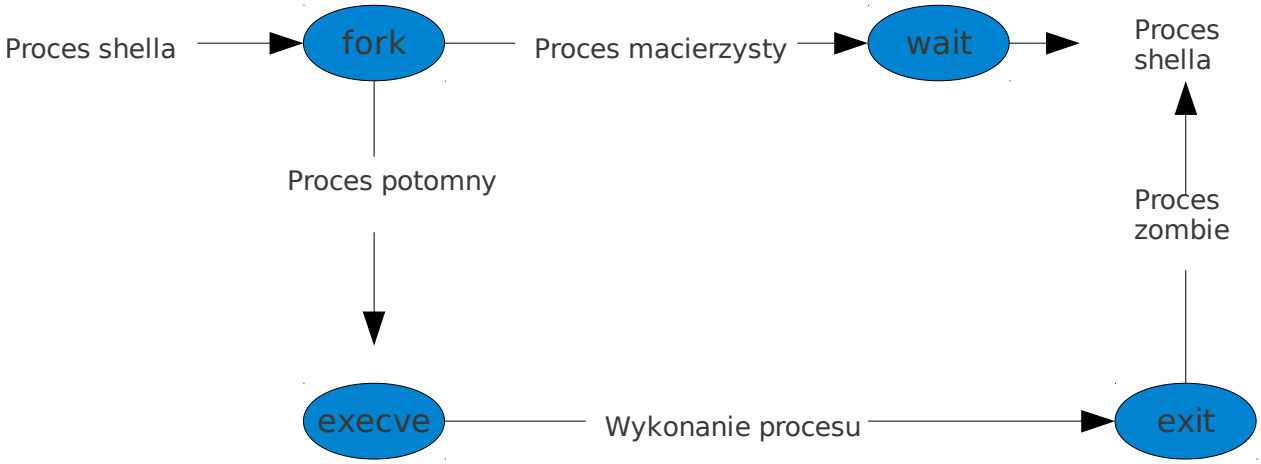
Wejściem dla systemu operacyjnego są programy – program, system może podzielić na jednostki zwane zadaniami (tasks) – wykonujące się zadanie (po przydzieleniu mu przez system czasu i przestrzeni) to proces.

W systemie UNIX procesy są rozróżniane za pomocą identyfikatorów będącymi liczbami całkowitymi:

- Wszystkie programy (systemowe i użytkownika) są wykonywane przez interpretera poleceń – shell – można dowolnie zastawiać programy jako scenariusze shell'a – tzw. pliki poleceń
- Wykonanie polecenia odbywa się za pomocą funkcji systemowych fork i execve odniesionych do pliku, możliwa jest praca pierwszoplanowa i drugoplanowa
- Procesy mogą dowolnie otwierać pliki – typowo w chwili rozpoczęcia są otwarte trzy pliki: standard wejście, standard wyjście, standard wyjście diagnostyczne
- Wszystkie procesy są potomkami jednego pierwotnego procesu init – którego identyfikatorem jest liczba 1
- Każdy aktywny terminal ma proces getty, zainicjowany przez proces init
- Proces getty określa początkowe parametry linii terminala i oczekuje na nazwę

otwierającą sesję z użytkownikiem, login – którą przekazuje za pomocą funkcji systemowej execve do procesu login

- Proces login pobiera hasło użytkownika, szyfruje je i porównuje z zaszyfrowanym napisem z pliku etc/passwd, jeśli porównanie wypada pomyślnie zezwala się użytkownikami na wejście do systemu
- Proces login rozpoczyna wykonywanie procesu shell – interpretera poleceń – ustanawiając identyfikator użytkownika procesu według danych rejestracyjnych użytkownika
- Nowy proces tworzy funkcja systemowa fork. Nowy proces zawiera kopię przestrzeni adresowej procesu pierwotnego



- Procesy mogą komunikować się między sobą poprzez łącza komunikacyjne – pipes – punkty końcowe procesów są zestawiane pomiędzy wykonaniem funkcji fork – a wykonaniem funkcji execve