Programowanie współbieżne i rozproszone – laboratorium

Instrukcja do ćwiczenia laboratoryjnego I

Procesy

Autor: S. Samolej

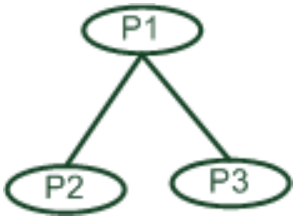
Rzeszów, 2013

Podziękowania:
Składam podziękowania panu dr inż. Jędrzejowi Ułasiewiczowi za udostepnienie materiałów wykładowych i laboratoryjnych, które były inspiracją do opracowania tych materiałów.

1. Proszę uruchomić terminal i przetestować działanie poleceń:
   ps – pobieranie informacji o aktywnych procesach
   ps –ef – wszystkie procesy w systemie
   ps – axjf – drzewo procesów
   ps – sLf - informacja o wątkach

   top – dynamiczna lista aktywnych procesów w systemie
   pstree – drzewo procesów w systemie

   Na podstawie instrukcji obsługi poleceń (np.: man ps) należy dowiedzieć się, jakie informacje zostają podane po wywołaniu podanych poleceń.
2. Proszę skompilować i uruchomić przykładowy program „system1.c".
   a. Należy przeanalizować wynik jego działania gdy poleceniem wywoływanym w funkcji system jest „ps –ax" oraz „ps-ax &".
   b. Należy opracować własny prosty program „Hello World" i uruchomić go z zastosowaniem funkcji system z innego programu.
3. Proszę skompilować i uruchomić przykładowy program „pexec1.c".
   a. Należy zmodyfikować program w taki sposób, aby wywołać w nim inne polecenie Linux.
   b. Należy zmodyfikować program w taki sposób, aby uruchomić z jego poziomu inny własny program („Hello World").
4. Proszę skompilować i uruchomić przykładowy program „fork1.c".
   a. Należy wydłużyć działanie obu procesów z programu i sprawdzić odpowiednimi instrukcjami (ps –ef |grep fork), czy w systemie pracują 2 procesy i jakie są ich parametry.
   b. Proszę zamienić ilość pętli wykonywanych przez procesy i wykryć z zastosowaniem odpowiednich poleceń systemowych, że jeden z procesów jest w stanie zombie.
5. Proszę skompilować i uruchomić przykładowy program „wait.c".
   a. Należy porównać działanie tego programu z działaniem programu „fork.c" i zwrócić uwagę na możliwość przechwycenia numeru PID procesu, z którym synchronizuje się proces macierzysty.
6. Proszę zaproponować program, który będzie miał 2 bezpośrednie procesy potomne jak na schemacie:



7. Proszę zaproponować program, który będzie tworzył kaskadę procesów potomnych jak na schemacie: