OpenSolaris

I . Sistemul de operare (SO) OpenSolaris este un sistem de operare cu codul sursă deschis, realizat de către compania Sun Microsystems pe baza SO Solaris . OpenSolaris este descendent a codului de bază de la UNIX System V Release 4, dezvoltat de către compania Sun şi AT&T în anii 1980.

Acest termen, OpenSolaris, este utilizat cu referire la codul sursă deschis a SO Solaris şi pentru a identifica comunitatea care a dezvoltat acest SO. După achiziţionarea companiei Sun Microsystems de către Oracle Corporation, în 2010, aceasta din urmă decide să întrerupă dezvoltarea deschisă a softului de nucleu, înlocuind OpenSolaris cu un model de distributiv cu titlu de proprietate a companiei Oracle, această versiune fiind Solaris Express.

Înainte de această achiziţie, Sun Microsystems a făcut publice codurile sursă pentru Solaris în data de 14 iunie 2005. Acestea includeau codurile sursă pentru nucleul SO, a componentelor de reţea, a bibliotecilor de sistem şi a programelor de bază pentru arhitecturile de procesoare SPARC şi x86 . Componentele date pot fi luate ca bază pentru crearea distributivelor de către întreprinderi terţe, lucru realizat atât cu participarea Sun Microsystems, cât şi fără implicarea acesteia.

La 19 martie 2007, Sun a anunțat că l-a angajat pe Ian Murdock, fondatorul Debian, pentru a conduce Project Indiana, un efort de a produce o distribuție completă OpenSolaris, cu instrumente GNOME și userland de la GNU, plus un pachet pentru management de sistem bazat pe rețea. Noua distribuție a fost planificată pentru a reîmprospăta experiența utilizatorului și a deveni succesorul Solaris Express ca bază pentru lansările viitoare ale Solaris.

II. 1 . OpenSolaris poate fi considerat ca fiind o extensie a masinii deoarece utilizează un sistem de gestionare a pachetelor numit Image Packaging System cunoscut și sub numele de pkg (5) pentru a adăuga, elimina și gestiona software-ul instalat și pentru a face actualizări la versiunile mai noi.

2.

3.Poate fi utilizat pe PC ,Mainframe,Multiprocesoare.

4.Are o structura monolitica deoarece dupa cum am mentionat si mai sus OpenSolaris este un descendent a codului de bază de la UNIX System V Release 4.

III 1

2 Atunci cant procesele sunt pregatite pentru a fi rulate ele se afla in starea “runnable in memory ” el fiind in memoria principal dar procesorul este ocupat , atunci cand unui process ii este asignat procesorul el se afla in starea “running” . Un process care se afla in memoria principal dar asteapta un anumit eveniment pentru executia sa el este in starea “sleeping in memory ” ,un proces care este situate in memoria secundara si asteapta un eveniment specific atunci el este in starea “sleeping and swapped ”. Daca un process se afla in memoria secundara si nu asteapta un eveniment specific el este in starea “runnable swapped”.

3 Principalele apeluri utilizate in acest sistem de operare pentru controlul proceselor sunt assignCPU,preempt , sleep , wakeup,swapin,swapout. Assign CPU asigneaza procesorul unui proces iar preempt trimite un proces din starea “running ” in starea “runnable in memory” sleep trimite un process din modul runnable in modul de asteptare wakeup face inversul .Apelurile lui swapin si swapout scot si introduce procese din si in pemoria principal.

4

IV 1 Sistemul de operare Open Solaris suporta FCE iar structura acestor fire de executie este pet trei nivele ULT user-level-thread , LWT lightweight process si kernel threads . Aceasta structura are scopul de a usura managmentul firelor de executie pentru sistemul de operare si de a oferi o interfata curata aplicatiilor .

2 Firele de executie folosesc o cantitate limitata de informatii in comparatie cu procesele , deasemenea intre firele de executie exista un ridicat grad de interdependenta pecand procesele sunt in competitie pentru resurse

3 Firele de executie in solaris pot fi in unul din cinci stari Stopped, Blocked, Run queue, Dispatchable sau  On LWP . Firele suspendate sunt in starea Stopped sau Blocked . Daca un fir este gata de executie el este in starea Run queue iar cande se gaseste un LWP pentru acest fir el trece in starea Dispatchable in final cand firul e preluat de catre LWP el se afla in starea On LWP.

4 In OpenSolaris se utilizeaza functia thr\_create() pentru a crea fire de executie noi . Functia start\_routine() este cea cu care incepe executia unui nou fir .Executia unui fir poate fi suspendata de catre Thr\_suspended() iar aceasta executie poate fi reluata cu ajutorul functiei thr\_continue . mai exista si functii precum thr\_exit() , thr\_detached , thr\_bound si thr\_daemon .

V 1. In Open Solaris nucleul implementeaza un model global bazat pe prioritate pentru firele de control a executiei din spatial nucleu .Planificatorul alege firul ce va fi executat in continuar bazandu-se pe acest model . Nucleul suporta notiunea de planificare preemtiva permitand intreruperea activitatii unui fir de control a executiei pentru a face loc unui fir cu o prioritate mai mare . Exista 170 de prioritati globale iar o valoare numaerica mai mare corespunde unei prioritati mai bune . Planificatorul din Open Solaris implementeaza diverse clase de planificare ce permit existenta mai multor politici de ordonare si alegere a firelor .

2. Dupa cum am mentionat si la punctul anterior in Open Solaris exista mai multe clase de planificare iar aceste sunt urmatoarele. TS (Timeshare Scheduling) esteclasa de baza pentru procese si firele din spatial nucleu . TS modifica in mod dinamic prioritatile bazanduse pe nivelul recent de utilizare al procesorului in incercarea de a distribui resursele in mod egala intre fire de executie . IA este o varianta mai complexa a TS ce mareste prioritatea firelor ce sunt legate de fereastra pe care se concentreaza utiizatorul . Atat TS cat si IA folosesc prioritati globala intre 0 si 59. FSS (Fair Share Scheduling) se bazeaza pe inpartirea resurselor procesorului in unitati iar firele sunt planificate pe baza acestor unitati si utilizarea procesorului .FX (Fixed - Priority) in cazul acesta firele nu isi modifica prioritatea . Mai exista si SYS ce utilizeaz ptioritati de la 60 la 99 si RT (Real time) cu prioritati de la 100 la 159.

3. In mod implicit procesele si firele de control al executiei vor fi incadrate in una din clasele TS sau IA . Exista comenzi pentru a muta fire si procese in alte clase precum prioctnl(1) . Este important de tinut minte ca pentru a mari prioritatile si pentru a folosi clasa RT eset necesar un utilizator privilegiat. Functia cl\_getclinfo este utilizata pentru a afla informatii despre clasa in care se afla firul sau procesul . Se utilizeaza si cl\_enterclass pentru alocarea resurselor necesare unei clase de prioritate si cl\_exitclass pentru a indeparta datele specifice unei clase .