UNIX I WONDOWS KAO RTOS

UNIX kao real-time operativni sistem

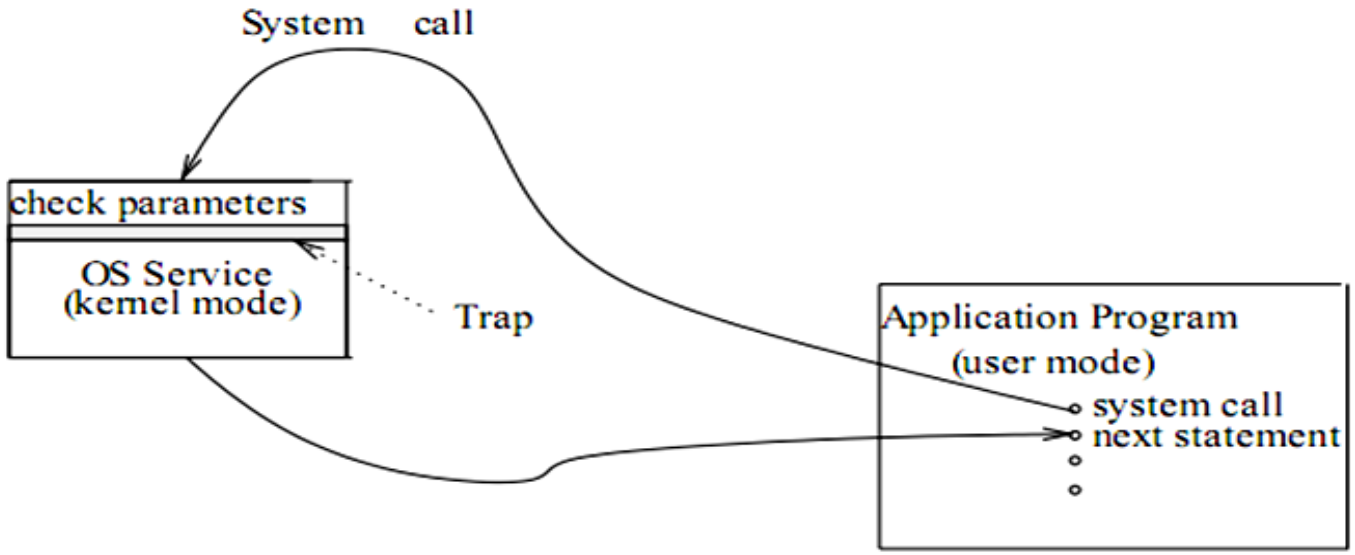
Obzirom da je Unix i njegove varijante široko rasprostranjen potrebno je istražiti kada Unix može biti korišten u real-time aplikacijama. Tradicionlani Unix operativni sistem pati od nekoliko nedostataka kada se koristi u real-time aplikacijama. Dva najčešča problema sa kojima se programer susreće kada koristi Unix za real-time aplikacije je ne priemptivni Unix kernel i dinamička promjena prioriteta taskovima.

Nepriemptivni krenel

Jedan od največih problema sa kojim se suočavaju real-time programeri dok koriste Unix za razvoj real-time aplikacije je taj da Unix kernel ne može biti priemptiran. To znači da su svi interapti onemogučeni kada se izvršava bilo koja rutina operativnog sistema. Aplikativni programi mogu pozvati servise operativnog sistema preko sistemskih poziva (System calls).

UNIX I WONDOWS KAO RTOS

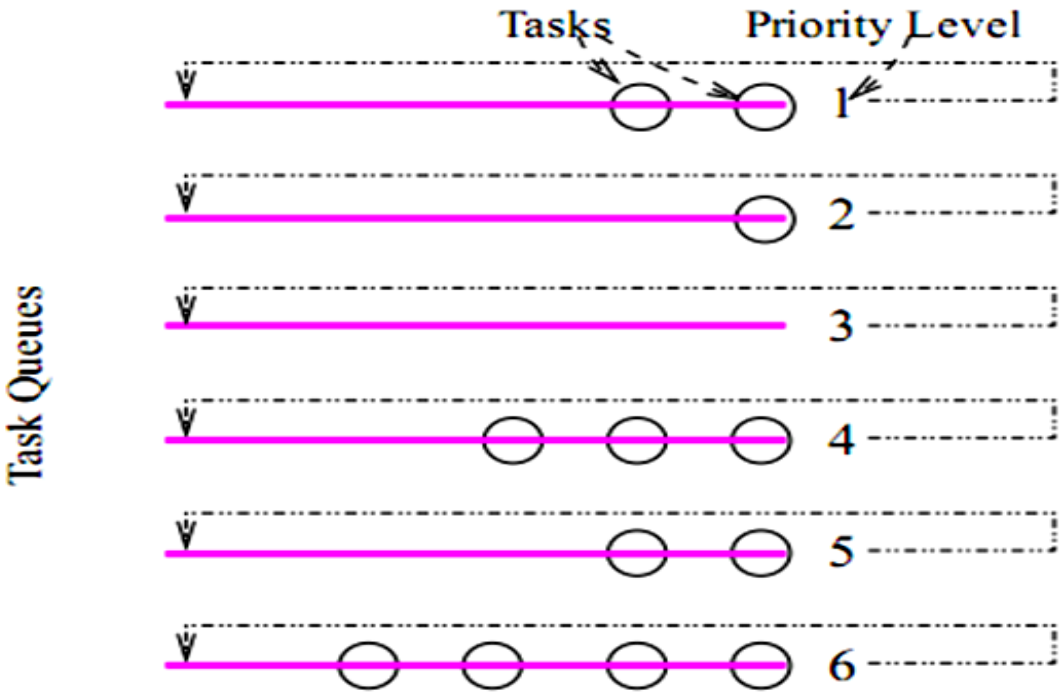
Primjeri sistemskih poziva uključuju servise operativnog sistema za kreiranje procesa, međuprocesnu komunikaciju, I/O operacije itd. Nakon što aplikacija pozove sistemski poziv, provjeravaju se argumenti predati od strane aplikacije koja obavlja sistemski poziv, kao sto je prikazano na slijedecoj slici:



Provjera argumenata od strane aplikacije koja obavlja sistemski poziv

UNIX I WONDOWS KAO RTOS

Dalje, izvršava se specijalna instrukcija nazvana trap (ili softverski interapt). Čim se izvršava trap instrukcija, rutina za rukovanje mijenja stanje procesora iz user moda u kernel mode (ili supervisor mode), i izvršavanje zahtijevane rutine kernela započinje. Promjena režima za vrijeme sistemskog poziva je shematski prikazana na narednoj slici:



Round-robin raspoređivanje sa multilevel feedback algoritmom

UNIX I WONDOWS KAO RTOS

Kompletne operacije kao što je upravljanje uređajima, kreiranje procesa, operacije sa fajlovima itd., treba da budu urađene samo u kernel modu. Na ovaj način kernel je u mogućnosti da primjenjuje disciplinu između različitih programa koji pristupaju ovim objektima. U slučaju da se ove operacije ne obavljaju u kernel modu, različite aplikacije mogu interferirati sa drugim aplikacijama. Primjer operativnog sistema kod kojeg su se sve operacije obavljale u user modu je nekada popularni operativni sistem DOS. Kod DOS-a, aplikacija je mogla slobodno obavljati bilo koju operaciju u user modu, uključujući rušenje sistema brisanjem sistemskih fajlova. Ovakva nestabilnost je neprihvatljiva u real-time okruženju. Tačnije, u DOS-u je bio samo jedan režim obavljanja operacija, user mode i kernel mod je neprimjetan.

UNIX I WONDOWS KAO RTOS

U Unix-u, proces koji se izvršava u kernel modu ne može biti zamijenjen (preempted) od strane drugog procesa. Drugim rječima, Unixov kernel je neprijemptivan. S druge strane, Unix sistem ne zamijenjuje procese koji se izvršavaju u user modu. Posljedica ovoga je da čak i kada proces niskog prioriteta napravi sistemski poziv, procesi visokog prioriteta će morati čekati dok se poziv napravljen od strane procesa niskog prioriteta ne kompletira. Za real-time aplikacije ovo uzrokuje inverziju prioriteta (priority inversion). Najduži sistemski pozivi mogu trajati i do nekoliko stotina milisekundi da se izvrše. Najgori slučaj vremena zamijene od nekoliko stotina milisekundi može lahko prouzrokovati da taskovi visokog prioriteta sa kratkim rokom za izvršenje (deadline) reda od nekoliko milisekundi promaše taj rok. Kod Unix-a, kada se kernel rutina počne da izvršava, svi interapti se onemogučuju.

UNIX I WONDOWS KAO RTOS

Interapti se uključuju samo nakon što se rutina operativnog sistema izvrši. Ovo je veoma efikasan način čuvanja integriteta strukture podataka kernela. Na ovaj se način smanjuju preopterećenja koja se javljaju pri postavljanju i uklanjanju brava (lock) i rezultira manjim prosječnim vremenom zamijene taska. Iako neprijemptivni kernel može dovesti do najgoreg vremena odgovora koje može iznositi i do sekunde, smatran je prihvatljivim od strane dizajnera Unix operativnog sistema. U to vrijeme, dizajneri Unixa nisu predviđali korištenje Unixa u real-time aplikacijama. Naravno, moguće je obaviti izmjene strukture podataka kernela korištenjem brava na odgovarajućim mjestima prije nego isključivati interapte, ali to bi rezultiralo povećanjem prosječnog vremena zamjene taska.

UNIX I WONDOWS KAO RTOS

Dinamički nivoi prioriteta

U tradicionalnim Unix sistemima, real-time taskovima se ne može pridružiti statička vrijednost prioriteta. Ubrzo nakon što programer postavi vrijednost prioriteta za task, operativni sistem je mijenja tokom izvršavanja taska. Ovo čini veoma teškim raspoređivanje real-time taskova korištenjem algoritama kao što su RMA ili EDF, jer oba ova raspoređivanja podrazumijevaju da jednom kada su dodijeljeni prioriteti tasku, ne bi trebali biti mijenjani od bilo kojeg dijela operativnog sistema. Poučno je razumijeti zašto Unix treba da dinamički mijenja vrijednosti prioriteta taskova. Unix koristi round robin raspoređivanje taskova sa multilevel feedback algoritmom. U ovoj shemi, raspoređivač reda taskove u višenivojske redove kao što je prikazano na prethodnoj slici. Na svakoj tački priempcije, raspoređivač skanira višenivojski red od vrha (najviši prioritet) i odabire task na vrhu prvog reda koji nije prazan

UNIX I WONDOWS KAO RTOS

Svakom tasku je dozvoljeno da se izvršava određeno vrijeme (time slice). Ukoliko se proces koji se izvršava blokira ili završi u okviru jedne sekunde tada se sklanja i raspoređivač odabire slijedeći task za izvršenje. Unix dopušta konfigurisanje podrazumijevane jedne sekunde za vremenski odsječak za vrijeme kreiranja operativnog sistema (system generation). Kernel izmješta proces koji se nije završio unutar dodjeljenog mu vremena, ponovo izračunava prioritet, i vraća ga u jedan od nekoliko redova prioriteta u zavisnosti od izračunate vrijednosti prioriteta taska.

Unix periodično proračunava prioritet taska na osnovou vrste taska i njegove istorije izvršavanja. Prioritet taska se ponovo izračunava na kraju njegovog j-tog odsječka vremena korištenjem slijedeća dva izraza:

UNIX I WONDOWS KAO RTOS

$$pri(T_i, j) = Base(T_i) + CPU(T_i, j) + nice(T_i)$$

$$CPU(T_i, j) = \frac{U(T_i, j-1)}{2} + \frac{CPU(T_i, j-1)}{2}$$

gdje je $pri(T_i, j)$ prioritet taska T_i na kraju njegovog j-tog odsječka vremena; $U(T_i, j)$ je upotreba taska T_i za j-ti odsječak vremena, i $CPU(T_i, j)$ je istorija upotrebe (weighted history) CPU-a taska T_i na kraju njegovog j-tog odsječka vremena. $Base(T_i)$ je osnovni prioritet taska T_i i $nice(T_i)$ je fina vrijednost dodijeljena T_i .

Korisnički procesi mogu imati negativnu nice vrijednost. Tako, nice vrijednost efektivno snižava vrijednost prioriteta procesa (npr. postaje fin sa drugim procesom).

Izraz je rekurzivno definisan. Odmotavanjem rekurzije dobijamo:

$$CPU(T_i, j) = \frac{U(T_i, j-1)}{2} + \frac{U(T_i, j-1)}{4} + \cdots$$

UNIX I WONDOWS KAO RTOS

Lahko se može vidjeti iz izraza da u izračunavanju istorije upotrebe CPU od strane taska, aktivnost (npr. procesiranje ili I/O) taska daje maksimalnu vrijednost. Ako task koristi CPU za cijelokupno vrijeme trajanja odsječka vremena (npr. 100% iskorištenost procesora), tada je $CPU(T_i, j)$ izračunat da bude visoka vrijednost – indicira smanjenje prioriteta taska.

S druge strane, ako je task blokiran zbog I/O operacije odmah nakon što je počeo tokom njegovog odsječka vremena, tada će $CPU(T_i, j)$ biti izračunat kao niska vrijednost, pokazujući povećanje prioriteta taska. Treba primjetiti da aktivnosti taska u prethodnim intervalima postaju progresivno niže. Treba biti jasno da $CPU(T_i, j)$ uzima istoriju upotrebe procesora za task T_i na kraju njegovog j-tog odsječka vremena.

Sada zamjenom izraza dobijamo:

$$pri(T_i, j) = Base(T_i) + \frac{U(T_i, j-1)}{2} + \frac{U(T_i, j-1)}{4} + \cdots + nice(T_i)$$

UNIX I WONDOWS KAO RTOS

Svrha pojma baznog prioriteta ($Base(T_i)$) u izrazu za izračunavanje prioriteta je da podijeli sve taskove u skup nivoa prioriteta sa fiksiranim granicama. Jednom kada je dodijeljen nivou prioriteta, nije moguće da se task premješta iz njemu dodijeljenih granica u granice drugih prioriteta zbog dinamičkog proračuna prioriteta. Vrijednosti $U(T_i, j)$ i $nice(T_i)$ komponenti su namjerno ograničene da budu dovoljno male da spriječe proces migracije iz njemu dodijeljenih granica.

Dinamičko proračunavanje prioriteta je motivisano zbog slijedečih razmatranja: Dizajneri Unix-a smatraju da je u bilo kojem računarskom sistemu, brzina prijenosa I/O uglavnom odgovorna za bilo kakvo sporo vrijeme odgovora. Procesori su ekstremno brzi u poređenju sa brzinom prijenosa I/O uređaja. Kašnjenje uzrokovano I/O transferima su uska grla u dostizanju bržeg odziva taska. Za ublažavanje ovog problema, poželjno je držati I/O kanale što je više moguće zauzetijim.

UNIX I WONDOWS KAO RTOS

Ovo se može postići dodijeljivanjem I/O taskovima većih prioriteta.

Kao što je već spomenuto, Unix ima skup prioriteta kojima su pridruženi različiti tipovi taskova. Različite granice prioriteta kod Unix-a prema opadajučem redoslijedu prioriteta su: swapper, block I/O, manipulacija fajlovima, kontrola tastature i uređaja i korisnički procesi. Taskovi koji obavljaju block I/O dodijeljeni su najvećem prioritetu. Ali, kada se zahtijeva block I/O? Da bi dali primjer block I/O, razmotrimo I/O koji se događa dok obrađujemo page fault u sistemu virtualne memorije. Block I/O koristi DMA-baziran transfer, i stoga čini efikasnom upotrebu I/O kanala. Karakter orjentisani I/O uključuje transfere od miša i tastature. Granice prioriteta su dizajnirane tako da pruže najefikasniju upotrebu I/O kanala. Da bi držali I/O kanale zauzetim, bilo koji task koji obavlja I/O operacije ne bi trebao biti držan da dugo čeka na CPU.

UNIX I WONDOWS KAO RTOS

Iz ovog razloga prioritet taska bi trebao biti povećan prema pravilu za izračunavanje prioriteta datom u Izrazu. Osnovna filozofija Unix operativnog sistema je da su interaktivni taskovi napravljeni da dostignu više nivoe prioriteta i budu što je prije moguče procesirani. Ovo korisnicima daje dobro vrijeme odziva. Ova tehnika je postala prihvačen način raspoređivanja mehkih real-time taskova širom većine dostupnih operativnih sistema opšte namjene, kao što je Microsoft Windows operativni sistem.

Gore razmatrane observacije daju ukupni efekat periodičnog izračunavanja vrijednosti prioriteta taska korištenjem Izraza kako slijedi:

Kod Unix operativnog sistema, dinamičko izračunavanje prioriteta uzrokuje da taskovi koji intenzivno koriste I/O operacije migriraju na nivoe višeg prioriteta, dok su taskovi koji intenzivno koriste CPU su na nižem nivou prioriteta.

UNIX I WONDOWS KAO RTOS

Skoro svaki moderni operativni sistem radi veoma sličan proračun prioriteta taska kako bi maksimizirali cjelokupnu propusnost sistema i pružili dobro prosječno vrijeme odgovora za interaktivne taskove. Međutim, za tvrde real-time taskove dinamičko izmjenjivanje vrijednosti prioriteta je očigledno neprikladno jer sprečava taskove da budu raspoređeni na nivoe višeg prioriteta, i također sprečava raspoređivanje pod popularnim real-time task algoritimima raspoređivanja kao što su EDF i RMA.

Ostali nedostatci Unix-a

Do sada je bilo riječi o dva glavna nedostatka Unix-a u rukovanju zahtjevima real-time aplikacija: dinamička izračunavanja prioriteta i neprijemptivni kernel. U nastavku ćemo spomenuti nekoliko drugih nedostataka Unix-a koji dolaze do izražaja kada pokušamo koristiti Unix u real-time aplikacijama.

UNIX I WONDOWS KAO RTOS

Nedovoljna podrška za drajvere. U Unix operativnom sistemu drajveri uređaja se izvršavaju u kernel modu. Dakle, ako dodamo novi uređaj, tada modul drajvera mora biti povezan sa modulom kernela – ovo zahtjeva ponovno kreiranje operativnog sistema (System generation). Kao rezultat, pružanje podrške za ovi uređaj u već razmještenu aplikaciju je komplikovano.

Nedostatak real-time fajl servisa. Kod Unix-a, fajl blokovi su alocirani kada se zahtijevaju od strane neke aplikacije. Kao posljedica toga, dok task piše u fajl, može doći do greške kada disk ostane bez slobodnog prostora. Drugim rječima, ne postoji garancija da će prostor na disku biti dostupan kada task upisuje blok u fajl. Tradicionalni pristupi upisa u fajl također rezultuje sporim pisanjem jer je potrebno najprije alocirati zahtijevani prostor prije upisa. Još jedan problem sa tradicionalnim fajl sistemima je da blokovi istog fajla moraju biti kontinualno

UNIX I WONDOWS KAO RTOS

poredani na disku. To bi rezultiralo da operacije čitanja traju nepredvidivo vrijeme, što za posljedicu ima duže vrijeme pristupa podacima. Kod real-time fajl sistema značajnije poboljšanje performansi se može ostvariti kontinualnim upisom fajlova na disk. Kako fajl sistem najprije alocira prostor, vremena za operacije čitanja i pisanja su predvidiva.

Neadekvatna podrška za vremenske servise. Kod Unix sistema, podrška za real-time tajmer je nedovoljna za mnoge tvrde real-time aplikacije. Rezolucija sata koja se pruža aplikacijama je 10 milisekundi što je pregrubo za mnoge tvrde real-time aplikacije.

UNIX I WONDOWS KAO RTOS

Real-time operativni sistemi zasnovani na Unixu

Obični Unix nije pogodan za korištenje kod tvrdih real-time aplikacija. U nastavku ćemo analizirati različite pristupe koje treba poduzeti kako bi Unix učinili podesnim za real-time aplikacije.

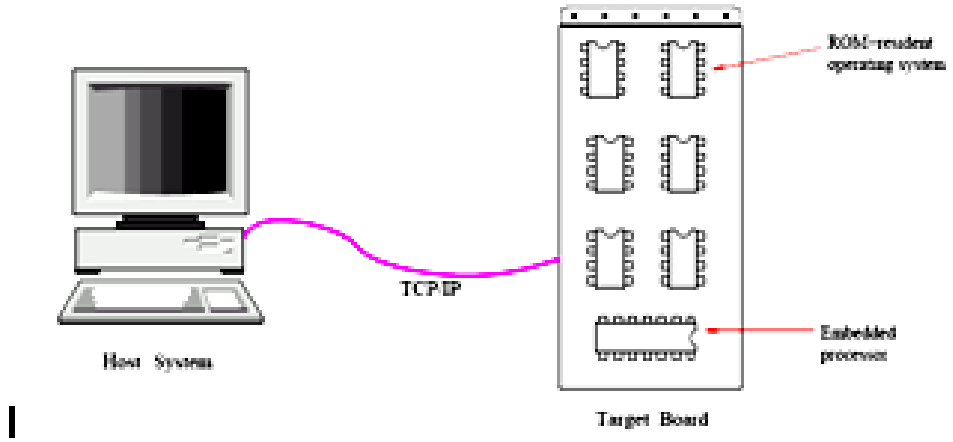
Dodaci tradicionalnom Unix kernelu

Naivan je pokušaj bio u prošlosti da se tradicionlani Unix napravi podesnim za real-time aplikacije dodavanjem nekih real-time mogućnosti preko osnovnog kernela. Ove dodatno implementirane mogućnosti su uključivale podršku za real-time tajmer, raspoređivač real-time taskova koji je izgrađen preko Unixovog raspoređivača, itd. Međutim, ova proširenja nisu riješila osnovne probleme sa Unix sistemom koji su ranije spomenuti; nepriemptivni kernel i dinamički nivoi prioriteta. Stoga ne čudi da površna proširenja mogućnosti Unix kernela bez rješavanja temeljnih nedostataka Unix sistema nisu zadovoljila zahtjeve tvrdih real-time aplikacija.

UNIX I WONDOWS KAO RTOS

Host-Target pristup

Host-target operativni sistemi su popularni u razvoju ugrađenih aplikacija. Kod ovog pristupa, razvoj real-time aplikacije se obavlja na host mašini. Host mašina je ili na Unix ili na Windows operativnom sistemu. Real-time aplikacija je razvijena na hostu a zatim je prebačena na ciljnu (target) ploču koja će da bude ugrađeni real-time sistem. U ROM ploče se upisuje mali kernel. Ovaj pristup je shematski prikazan na narednoj slici:



Shematski prikaz upisa kernela u ROM

UNIX I WONDOWS KAO RTOS

Host-Target pristup

Glavna ideja ovakvog pristupa je da se real-time operativni sistem koji se izvršava na ciljnoj ploči drži što je moguće više malim i jednostavnim. Ovo implicira da operativini sistem na ciljnoj ploči nema podršku za virtualnu memoriju, niti za bilo kakve alate kao što su kompajleri, programski editori, itd. Procesor na ciljnoj ploči će da pokreče real-time operativini sistem.

Host sistem je Unix ili Windows sistem koji podržava razvojno okruženje, uključujući kompajlere, editore, biblioteke, cross kompajlere, debuggere itd. Ovo su aplikacije koje zahtjevaju podršku sa virtualnom memorijom. Host je obično spojen na ciljni sistem korištenjem serijskog porta ili TCP/IP veze.

Real-time program se razvija na hostu. Zatim se cross-kompajlira kako bi se generisao kod za ciljni procesor. Zatim se izvršni modul prebacuje na ciljnu ploču. Taskovi se izvršavaju na ciljnoj ploči i izvršavanje se kontroliše na hostu korištenjem simboličkog cross debuggera.

UNIX I WONDOWS KAO RTOS

Jednom kad program uspješno proradi, on se uprži na ROM ili flash memoriju i postaje spreman za korištenje u aplikaciji za koju je razvijen.

Komercijalni primjeri host-target operativnih sistema uključuju PSOS, VxWorks i VRTX. O ovim operativnim sistemima će biti riječi nešto kasnije sa posebnim osvrtom na VxWorks. Cilj je da se prikaže kako target operativni sistemi obzirom na njihovu malu veličinu, ograničenu funkcionalnost i optimalan dizajn postižu daleko bolje performanse nego kompletni operativni sistemi. Na primjer, zamjena taska (task preemption) kod ovih sistema je reda nekoliko milisekundi u poređenju sa nekoliko stotina mili sekundi kod tradicionalnih Unix sistema.

UNIX I WONDOWS KAO RTOS

Korištenje tačaka priempcije

Već je spomenuto da je jedan od glavnih nedostataka tradicionalnog Unix V koda, je uzrokovan činjenicom da su za vrijeme sistemskog poziva svi interapti maskirani (onemogučeni) cijelo vrijeme dok traje izvršavanje sistemskog poziva. Ovo vodi ka neprihvatljivom vremenu odziva koje je reda od nekoliko sekundi čineći na taj način Unix bazirane sisteme neprihvatljivim za večinu tvrdih real-time aplikacija.

Jedan od pristupa koje su preduzeti od strane nekoliko proizvođača sa ciljem da unaprijede real-time performanse nepriemtivnih kernela je uvođenje tačaka zamjene (preemption points) u sistemske rutine. Tačke priempcije pri izvršavanju sistemske rutine su trenuci u kojima je struktura podataka kernela konzistentna. U takvim tačkama, kernel može sigurno biti priemptiran kako bi se napravilo mjesta za bilo koji real-time task na čekanju višeg prioriteta kako bi se izvršio bez

UNIX I WONDOWS KAO RTOS

ometanja bilo kakve strukture podataka kernela. Kod ovog pristupa, kada izvršavanje sistemskog poziva dođe do tačke priempcije, kernel provjerava da vidi da li je neki task većeg prioriteta postao spreman (ready). Ako postoji barem jedan, on izmješta kernel rutinu i hitno šalje task najvećeg prioriteta koji čeka. Najgora moguča latencija priempcije kod ove tehnike je najduže vrijeme između dvije uzastopne tačke priempcije. Kao rezultat, najgore vrijeme odziva taskova se smanjuje u poređenju sa onim kod tradicionalnih operativnih sistema bez tačaka priempcije. Ovo čini ove operativne sisteme (preemtion point-based operating systems) podesnim za korištenje u mnogim kategorijama tvrdih real-time aplikacija, iako još uvijek zaostaju za zahtjevima tvrdih real-time aplikacija koje zahtijevaju da latencija priempcije bude reda nekoliko mikrosekundi ili manje.

UNIX I WONDOWS KAO RTOS

Još jedna prednost ovakvog pristupa je da iziskuje minimalne promjene koje trebaju biti napravljene nad kodom kernela. Mnogi operativni sistemi su iskoristili ovaj pristup. Poznati komercijalni primjeri ovakvog pristupa uključuju HP-UX i Windows CE.

Self-Host sistemi

Za razliku od host-target sistema kod kojih se razvoj aplikacije obavljao na odvojenoj host mašini, kod self-host sistema real-time aplikacija je razvijana na istom sistemu na kojem će se real-time aplikacija na poslijetku izvršavati. Naravno, dok izvršavamo aplikaciju, moduli operativnog sistema koji nisu bitni za izvršavanje taska su isključeni kako bi se minimizirala veličina operativnog sistema. Treba zapamtiti da kod host-target pristupa, ciljni real-time operativni sistem je mršav i efikasan sistem koji može samo izvršavati aplikaciju ali

UNIX I WONDOWS KAO RTOS

ne može sadržavati mogućnosti za razvoj; razvoj programa se obavlja na host sistemu. . Ovo čini razvoj i debagiranje aplikacije teškim i zahtjeva podršku za cross-kompajler i cross-debugger. Self-host sistemi imaju drugačiji pristup. Real-time aplikacija je razvijena na samostalnom operativnom sistemu. Jednom kada se bez problema aplikacija izvrši na hostu i ako smo zadovoljni, ona se onda prži na ROM ili flash memoriju na ciljnoj hardverskoj ploči zajedno sa ogoljenom verzijom operativnog sistema.

Većina trenutno dostupnih self-host operativnih sistema, bazirani su na mikro-kernel arhitekturi. Upotreba mikro-kernel arhitekture za self-host operativni sistem podrazumijeva nekoliko prednosti. U mikro-kernel arhitekturi, implementirane su samo bitne funkcionalnosti kao što je rukovanje interaptima i upravljanje procesom kao rutine kernela.

UNIX I WONDOWS KAO RTOS

Sve ostale funkcionalnosti kao što je upravljanje memorijom, fajlovima, uređajima, itd. je implementirano kao moduli dodatci koji rade u korisničkom režimu.

Dodati moduli mogu biti lahko isključeni kad god nema potrebe za njima. Kao rezultat toga, konfiguracija operativnog sistema je veoma jednostavna, a to rezultira malim sistemom. Također, mikro kernel je mršav i zbog toga je dosta efikasniji u poređenju sa monolitnim. Druga poteškoča sa monolitnim operativnim sistemom je da povezuje većinu drajvera, fajl sistema i protokola sa kernelom operativnog sistema i svi kernel procesi dijele isti adresni prostor. Stoga jedna programerska greška u bilo kojoj od ovih komponenti može izazvati fatalnu grešku na kernelu. Kod operativnih sistema zasnovanih na mikro-kernelu, ove komponente se izvršavaju u odvojenim memorijskim prostorima.

UNIX I WONDOWS KAO RTOS

Zbog toga su padanja sistema veoma rijetka i operativni sistemi sa mikro-kernelom su veoma pouzdani.

U nastavku će biti riječi o prevazilaženju ovih problema kod self-host sistema.

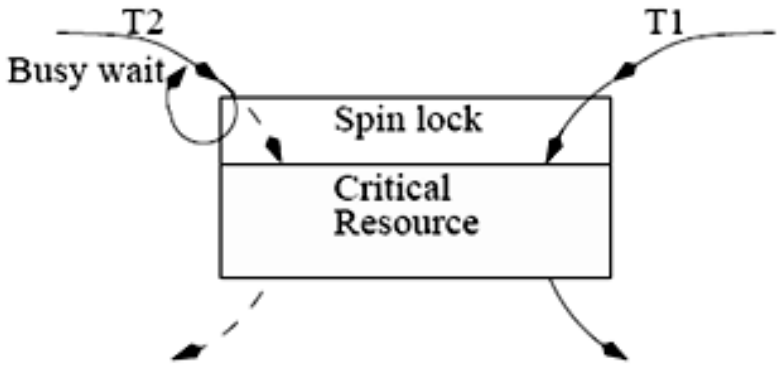
Nepriemptivni kernel. Sa ciljem da se sačuva integritet strukture podataka kernela, svi interapti su onemogučeni dokle god se sistemski poziv ne izvrši. Ovo je urađeno zbog efikasnosti i radi dobro kod jednoprocesorskih aplikacija koje nisu real-time. Maskiranje interapta za vrijeme dok kernel poziva čak i veoma male kritične rutine ima najgore vrijeme odziva reda sekunde. Dalje, ovaj pristup ne radi u multiprocesorskim okruženjima. U multiprocesoriskim okruženjima maskiranje interapta za jedan procesor ne pomaže u osiguranju integriteta strukture podataka kernela, pošto se taskovi izvršavaju na drugom procesoru koji opet može ugroziti strukturu podataka kernela.

UNIX I WONDOWS KAO RTOS

Neophodno je koristiti brave na odgovarajućim mjestima u kodu kernela kako bi se prevazišao problem. Fundamentalno pitanje u vezi zaključavanja je u trenutcima kada je potrebno obezbijediti sinhronizaciju u određenim segmentima koda u kernelu. Ovi segmenti se nazivaju kritične sekcije. Kao primjer problema, razmotrimo slučaj sa dva procesa gdje svaki proces treba da inkrementira vrijednost dijeljene varijable **i**. Pretpostavimo da jedan proces čita **i**, a zatim to radi drugi proces. Obadva procesa inkrementiraju **i**, zatim upisuju **i** u memoriju. Ako je na početku **i** bilo jednako 2, sada će biti jednako 3, umjesto da iznosi 4 !

Trebalo bi biti jasno da sa ciljem da se kernel učini priemptivnim, zaključavanja moraju biti iskorištena na odgovarajućim mjestima u kodu kernela.

UNIX I WONDOWS KAO RTOS



Spin zakljucavanje

Kod u potpunosti priemptivnih Unix sistema obično se koriste dvije vrste zaključavanja: zaključavanje na nivou kernela (kernel-level locks) i spin zaključavanje. Kernel-level zaključavanje je slično tradicionalnom zaključavanju. Kad task ćeka da kernel-level brava bude otključana od strane taska koji je drži zaključanom, on je blokiran i prolazi kroz izmjenu konteksta. On postaje spreman (ready) samo nakon što zahtijevana brava bude otključana od strane taska koji je drži zaključanom i kada brava postane dostupna .

UNIX I WONDOWS KAO RTOS

Kernel-level brave su neefikasne kada su potrebni resursi za neko kratko vrijeme koje je u poređenju sa vremenima izmjene konteksta reda nekoliko mili sekundi ili čak i manje. U takvim situacijama, dodatni posao izmjene konteksta može prekomjerno povećati vremena odgovora taska što je neprihvatljivo. Razmotrimo situaciju da neki task zahtijeva bravu da bi obavio veoma malo procesiranje (vjerovatno jednu aritmetičku operaciju) nad nekim kritičnim resursom koji će zahtijevati, recimo 1msec procesiranja. Sada, ako je kernel-level brava u upotrebi, drugi task drži bravu u vrijeme kada prvi task zahtjeva, task će biti blokiran i trebalo bi doći do izmjene konteksta, također i cache konteksta, stranice taska itd. mogu biti zamijenjeni. Nakon kratkog vremena, recimo 1msec, brava postaje dostupna. Može nastati druga izmjena konteksta uz pretpostavku da je to task sa najvišim prioritetom u tom trenutku. Stoga, vrijeme odgovora za T_i bi trebalo biti 3 msec.

UNIX I WONDOWS KAO RTOS

U ovakvim situacijama je neophodno spin zaključavanje.

Spin zaključavanje je shematski prikazano na prethodnoj slici. Kritični resurs se traži od oba taska T_1 i T_2 za veoma kratko vrijeme (u poređenju sa vremenom potrebnim za izmjenu konteksta). Ovaj resurs je zaštičen sa spin zaključavanjem. Pretpostavimo slijedeće, task T_1 je stekao spin zaključavanje kako bi zaštitio resurs, i u međuvremenu, task T_2 zahtijeva resurs. Kako je task T_1 zaključao resurs, T_2 ne može dobiti pristup resursu i onda on čeka (prikazano kao petlja na slici) i ne blokira i ne koristi izmjenu konteksta. T_2 dobija resurs ubrzo nakon što ga T_1 oslobodi. U višeprocesorskom sistemu, spin zaključavanja su obično implementirana korištenjem protokola za koherenciju keš memorije, tako da svaki procesor radi na lokalnoj kopiji lock varijable.

.

.

UNIX I WONDOWS KAO RTOS

Razmotrimo kako spin zaključavanje može biti implementirano na jednoprocesorskom sistemu. Na jednoprocesorskom sistemu, kada se zahjteva kritični resurs na veoma kratko vrijeme, uzajamno isključivanje (mutual exclusion) je jednostavno obavljeno onemogučavanjem interapta. Na Intel x86 sistemima ovo se uradi sa „**cli**“ instrukcijom. Interapti se ponovo omogućuju sa „sti“. Prema tome, spin zaključavanja na jednom procesoru je kompajlirano kao pozivi za „**cli**“ i „**sti**“.

Real-time prioriteti. Kod na Unix-u zasnovanih real-time operativnih sistema, kao dodatak dinamičkim prioritetima, podržani su real-time i idle prioriteti. Naredna slika shematski prikazuje tri dostupna nivoa prioriteta.

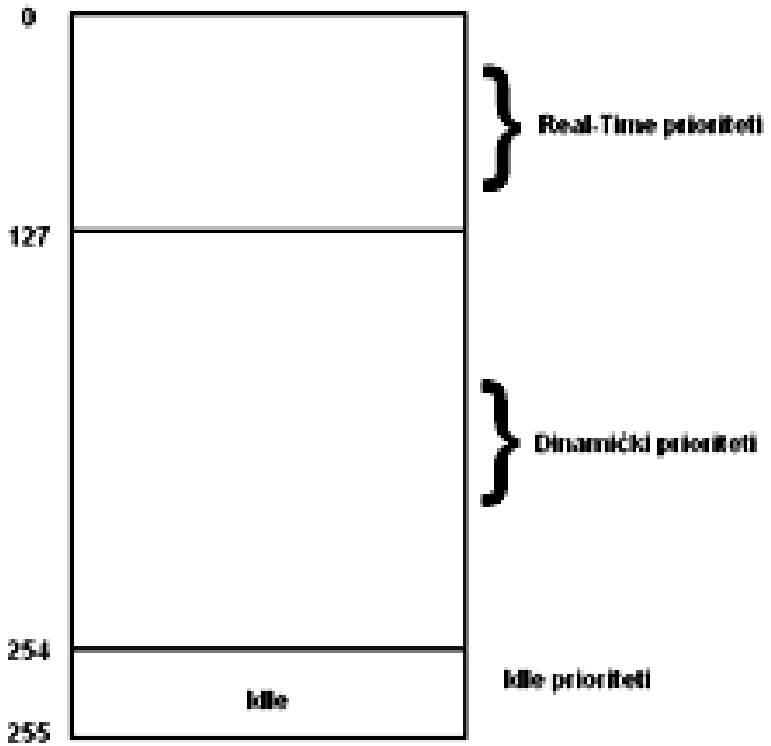
Idle (nemigracijski). Ovo je najniži nivo prioriteta. Task koji se izvršava kada nema drugog taska (idle task), izvršava se na ovom nivou. Idle prioriteti su statički i ne izračunavaju se periodično.

UNIX I WONDOWS KAO RTOS

Dinamički. Dinamički prioriteti se izračunavaju periodično kako bi poboljšali prosječno vrijeme odziva mehkih real-time (interaktivnih) taskova. Dinamički izračun prioriteta osigurava da I/O taskovi migriraju na veće prioritete i CPU taskovi rade na nižim prioritetima. Kao što je prikazano na narednoj slici, taskovi na dinamičkom nivou prioriteta rade sa prioritetima višim od idle prioriteta, ali na nižem prioritetu od real-time prioriteta.

Real-Time. Real-time prioriteti su statički prioriteti i ne izračunavaju se tokom izvršavanja aplikacije. Tvrdi real-time taskovi operiraju na ovim nivoima. Taskovi koji imaju real-time prioritete dobijaju prednost nad taskovima sa dinamičkim nivoima prioriteta. Shematski prikaz tri nivoa prioriteta je pokazan na narednoj slici.

UNIX I WONDOWS KAO RTOS



Shematski prikaz tri nivoa prioriteta taskova

UNIX I WONDOWS KAO RTOS

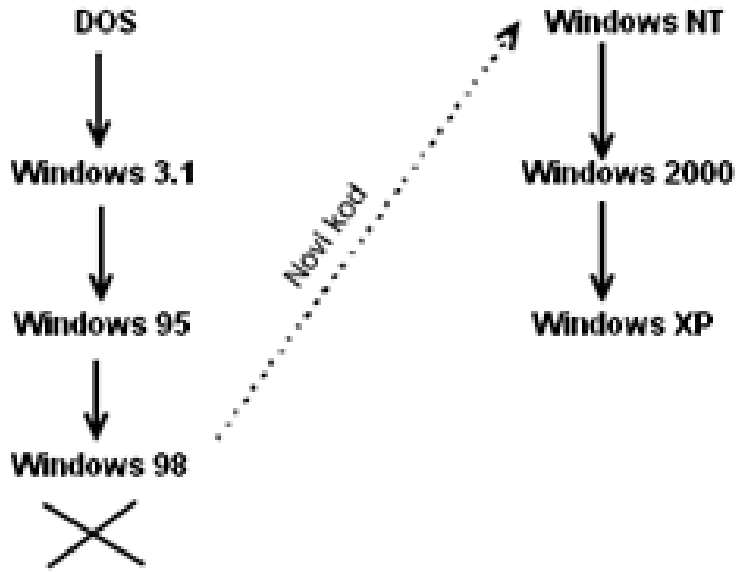
Windows kao Real-time operativni sistem

Operativni sistem Windows je evoluirao u poljednjih dvadeset pet godina od jednostavnog DOS-a (Disk Operating System) kojeg je Microsoft razvio u ranim osamdesetim. DOS je veoma jednostavan operativni sistem, koji podržava jedan task i koristi segmentiranu memorijsku shemu. U kasnim osamdesetim, DOS je evoluirao u Windows seriju operativnih sistema. Glavna razlika između DOS-a i Windowsa je grafički interfejs. 1998 godine, kôd Windowsa je u potpunosti izmjenjen i razvijen je Windows NT sistem. Obzirom da je kôd u potpunosti izmjenjen, Windows NT je daleko stabilniji nego ranije verzije bazirane na DOS-u. Kasnije verzije Microsoftvih operativnih sistema su nasljednici Windows NT operativnog sistema. Naredna slika prikazuje genealogiju različitih Microsoftovih operativnih sistema. Kao što je već spomenuto, stabilnost prethodnika Windows NT operativnog sistema nije bila zadovoljavajuća.

UNIX I WONDOWS KAO RTOS

Računarski sistemi zasnovani na Windows NT operativnom sistemu i njegovim nasljednicima masovno se koriste u kućama, kancelarijama i industriji. Sistem zasnovani na Windows NT operativnom sistemu mogu biti interesantni za upotrebu u real-time aplikacijama iz razloga jeftinije cijene i drugih pogodnosti. Ovo je naročito tačno kod razvoja prototipa aplikacije i tamo gdje se zahtjeva ograničena količina softvera. U nastavku ćemo se kritički osvrnuti na pogodnost korištenja Windows NT operativnog sistema za razvoj real-time aplikacija. Najprije ću se dotači nekih osobina Windows NT operativnog sistema koje su veoma bitne i korisne za razvoj real-time aplikacije. Također, biće riječi i o nedostatcima ovog operativnog sistema kada se koristi za razvoj real-time aplikacija.

UNIX I WONDOWS KAO RTOS



Genealogija razvoja Microsoft operativnih sistema

Bitne osobine Windows NT OS-a

Windows NT ima nekoliko osobina koje su veoma poželjne za real-time aplikacije kao što je podrška za višenitnost (multithreading), i dostupnost real-time nivoa prioriteta. Također, rezolucije tajmera i sata su dovoljno fine za većinu real-time aplikacija

UNIX I WONDOWS KAO RTOS

Windows NT podržava 32 nivoa prioriteta. Svaki proces pripada jednoj od slijedećih klasa prioriteta: idle, low, below normal, normal, above normal, high, real-time. Po defaultu, klasa prioriteta na kojem se izvršavaju korisnički taskovi je normal. Normal i high klase prioriteta su varijabilnog tipa, u smislu da se prioriteti taskova izračunavaju periodično od strane operativnog sistema. NT snižava prioritet taska ako koristi sve svoje posljednje odsječke vremena. On podiže prioritet taska ako je isti blokiran zbog I/O operacije i ne može koristiti u potpunosti svoj posljednji odsječak vremena. Međutim, izmjena taska od njegovog osnovnog prioriteta je ograničena na ±2. NT koristi prioritetom vođeno priemptivno raspoređivanje i real-time niti imaju prednost nad svim drugim nitima uključujući i niti kernela. Procesi kao što je screen saver koriste klasu prioriteta idle.

UNIX I WONDOWS KAO RTOS

Mane Windows NT OS-a

Unatoč impresivnoj podršci koju Windows NT pruža za real-time programiranje, programer koji želi da koristi Windows NT u real-time razvoju mora da se suoči sa nekoliko problema. Od tih problema, slijedeća dva problema su najkritičnija.

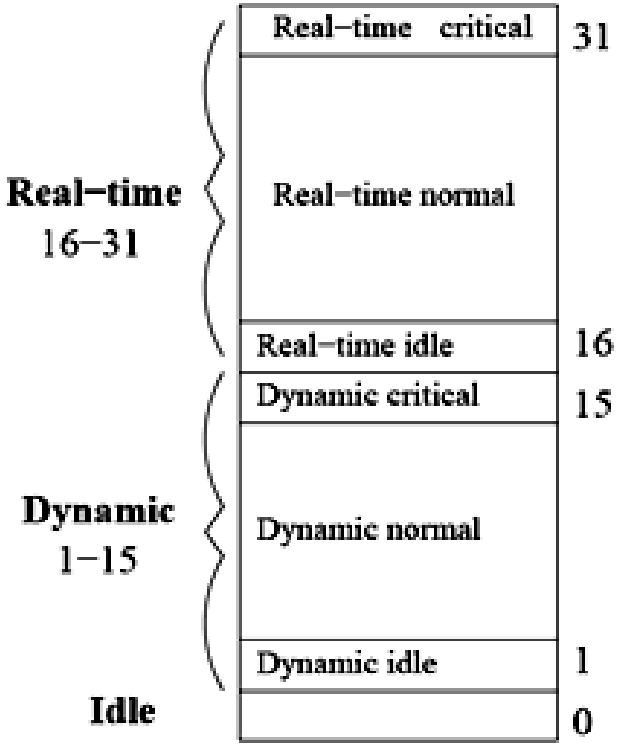
Obrada interapta. Kod Windows NT-a, nivo prioriteta interapta je uvijek veći nego kod korisničkih niti; uključujući niti real-time klase. Kada se desi interapt, rutina za obradu interapta sačuva stanje mašine i čini da sistem izvrši servisnu rutinu prekida (Interrupt Service Routine). Samo veoma kritično procesiranje se obavlja u servisnoj rutini prekida i masa procesiranja se obavi kasnije na nižem prioritetu u formi DPC-a (Diferred Procedure Call). Za različite interapte DPC-ovi se stavljaju u FIFO red čekanja.

UNIX I WONDOWS KAO RTOS

Ovo razdvajanje ISR-a i DPC-a ima prednost u osiguravanju brzog odgovora za buduće interapte, ali ima nedostatak u održavanju svi DPC-a na istim vrijednostima prioriteta. DPC ne može biti priemptiran od strane drugog DPC-a ali može biti priemptiran od strane interapta. DPC-ovi se izvršavaju po FIFO redoslijedu sa prioritetom nižim od prioriteta hardverskog interapta ali večim od prioriteta raspoređivača/dispečera. Ne postoji mogućnost za user-level niti da se izvrše sa prioritetom večim od prioriteta ISR-a i DPC-a. Dakle, čak i ISR-ovi i DPC-ovi koji odgovaraju taskovima veoma niskog prioriteta mogu izmjestiti real-time procese. Kao rezultat toga, potencijalno blokiranje real-time taskova od strane DPC-a može biti veliko. Na primjer, interapt koji se desi usljed greške stranice generisan od strane taska niskog prioriteta trebao bi se izvršiti brže od real-time procesa.

UNIX I WONDOWS KAO RTOS

Također, ISR-ovi i DPC-ovi generisan usljed pritiska tipke na tastaturi ili mišu će raditi na većem nivou prioriteta u poređenju sa real-time taskovima.



Windows nivoi prioriteta

UNIX I WONDOWS KAO RTOS

Dakle, u prisustvu procesa koji obrađuje mrežni ili disk I/O, efekt DPC FIFO reda čekanja može voditi do neograničenih vremena čekanja čak i kod real-time niti. Ovaj problem je izbjegnut kod Windows CE operativnog sistema kroz upotrebu mehanizma nasljeđivanja prioriteta.

Podrška za protokole za dijeljenje resursa (Resource sharing protocols).

Windows NT ne pruža bilo kakvu podršku (kao što je nasljeđivanje prioriteta itd.) u cilju podrške real-time taskovima kako bi dijelili kritične resurse između sebe. Ovo je glavni nedostatak Windows NT operativnog sistema kada se koristi u real-time aplikacijama. Kako većina real-time aplikacija zahtijevaju dijeljenje kritičnih resursa između taskova, najjednostavniji pristup je da dopustimo real-time taskovima da dijele kritične resurse. Da bi izbjegli inverzije prioriteta potrebno je pažljivo podešavanje prioriteta tokom uzimanja i oslobađanja brave.

UNIX I WONDOWS KAO RTOS

Čim je task uspješno obavio zaključavanje nepriemptivnog resursa, njegov prioritet može biti dinamički povećan. Međutim, poznato je da ovakvo uređenje može voditi do velikih inverzija vezanih za naslijeđivanje.

Još jedna opcija je da se implementira PCP (Priority Ceilling Protocol). Da bi implementirali ovaj protokol, treba da ograničimo real-time taskove da imaju samo parne prioritete (npr. 16,18,...,30). Razlog ovakve restrikcije je da Windows NT ne podržava FIFO raspoređivanje između taskova jednakog prioriteta. Ako je najviši prioritet među taskovima koji trebaju resurs 2*n, tada će prioritet resursa biti 2*n+1. Kod Unix-a, dostupna je mogućnost FIFO raspoređivanja između taskova jednakog prioriteta, tako da se mogu koristiti svi dostupni nivoi prioriteta kako bi se dodijelili taskovima.

UNIX I WONDOWS KAO RTOS

Windows NT nasuprot Unix operativnog sistema

Poređenje osnovnih osobina koje se zahtijevaju za real-time programiranje a koje su omogućene kod Windows NT i Unix V operativnih sistema prikazano je u narednoj tabeli. Može se vidjeti da Windows NT podržava daleko finiju granulaciju vremena (1mSec) u poređenju sa Unix-om (10mSec). Windows NT ima dosta osobina koje su poželjne za real-time operativni sistem. Međutim, način na koji izvršava DPC-ove, zajedno sa nedostatkom podrške protokola za dijeljenje resursa između taskova jednakog prioriteta čini ga nepodesnim za upotrebu kod safety-critical tvrdih real-time aplikacija. Sa pažljivim programiranjem, Windows NT može biti uspješno korišten za aplikacije koje mogu tolerisati povremeno probijanje zadanog roka (deadline-a) i koje imaju rok koji je reda od nekoliko stotina milisekundi .

UNIX I WONDOWS KAO RTOS

Naravno, da bi bio korišten u ovakvim aplikacijama, iskorištenost procesora mora biti dovoljno niska i kontrola inverzije prioriteta mora biti obezbjeđena na nivou korisnika.

Real-Time Osobine	Windows NT	Unix V
DPC-ovi	Da	Ne
Real-time prioriteti	Da	Ne
Zaključavanje virtualne memorije	Da	Da
Preciznost tajmera	1 mili sekunda	10 mili sekundi
Asinhroni I/O	Da	Ne

Osnovne real-time osobine Windows NT i Unix V operativnih sistema

Iako su Windows bazirani sistemi popularni za desktop aplikacije i pružaju većinu poželjnih osobina zahtijevanih za real-time programiranje, sistemi bazirani na Unix-u su daleko popularniji za real-time aplikacije obzirom na cijenu i „hakerski mentalitet “ real-time programera .

UNIX I WONDOWS KAO RTOS

Pregled savremenih Real-time operativnih sistema

U nastavku ćemo navesti neke od osobina danas popularnih real-time operativnih sistema koji se koriste u komercijalnoj primjeni. Studiranjem osobina komercijalno dostupnih real-time operativnih sistema može nam dati ideju o tome koji real-time operativni sistem koristiti za specifičnu real-time aplikaciju.

Prije nego što damo pregled nekih popularnih komercijalnih real-time operativnih sistema, napomenimo da mnogi od ovih operativnih sistema dolaze sa skupom alata kako bi se olakšao razvoj real-time aplikacija. Pored programerskih alata kao što su editori, kompajleri, dibageri, postoji i nekoliko naprednih alata posebno dizajniranih da pomognu razvoju real-time apliakcija. Alati uključuju analizatore memorije, analizatore performansi, simulatore itd.

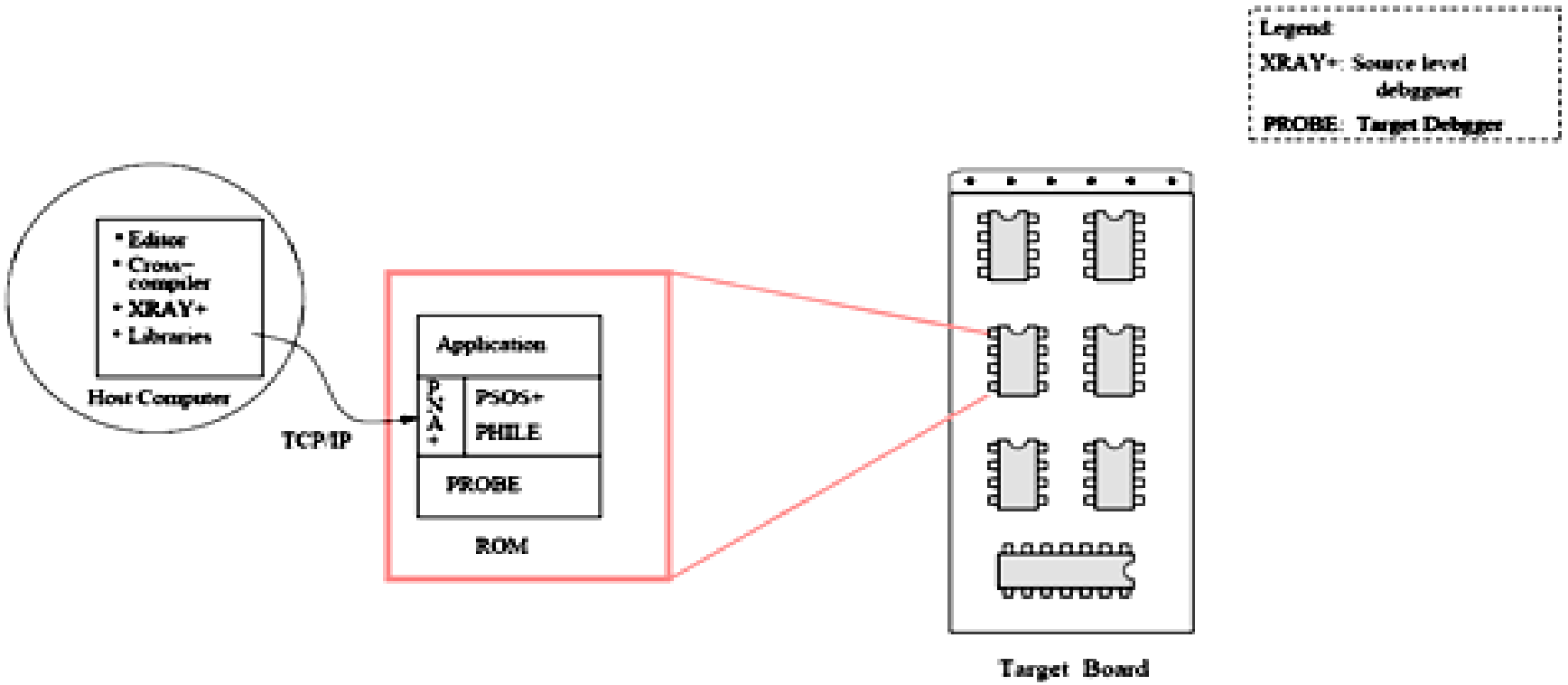
UNIX I WONDOWS KAO RTOS

PSOS

PSOS je popularan real-time operativni sistem koji se isključivo koristi za ugrađene (embedded) aplikacije. Proizvodi ga Wind River Systems, veliko ime u areni real-time operativnih sistema. PSOS je host-target tip real-time operativnog sistema i koristi se u nekoliko komercijalnih proizvoda. Primjer jedne primjene PSOS-a je u baznoj stanici za mobilnu telefoniju.

Razvoj PSOS aplikacije je shematski prikazan na narednoj slici. Host računar je obično desktop mašina. Podržani su i Windows i Unix host operativni sistemi. Ciljna ploča sadrži ugrađeni procesor, ROM, RAM, itd. Host računar pokreće editor, cross-kompajler, source-level debugger, biblioteku rutina. PSOS+ i drugi opcioni moduli kao što su PNA+, PHILE i PROBE su instalirani u ROM-u na ciljnoj ploči. PNA+ je mrežni menager.

UNIX I WONDOWS KAO RTOS



Razvoj PSOS aplikacije

On omogućava TCP/IP komunikaciju između hosta i targeta (ciljnog hardvera) preko Etherneta i FDDI (Fiber Distributed Data Interface). Usklađen je sa Unix (BSD) sokit sintaksom i kompatibilan je sa drugim TCP/IP baziranim mrežnim

UNIX I WONDOWS KAO RTOS

standardima kao što su FTP i NFS (Network File System). Njihovim korištenjem, PNA+ pruža efikasnu komunikaciju između hosta i targeta. PROBE+ je target debugger i XRAY+ je debugger na nivou koda. XRAY+ poziva PROBE+ kako bi pružio debugging okruženje real-time programeru. Razvoj aplikacije se obavlja na host mašini a zatim se prebacuje na ciljnu ploču. Aplikacija se debuggira korištenjem XRAY+ debuggera. Tokom razvoja aplikacije, aplikacija se prebacuje u RAM na ciljnoj ploči. Kada jednom zadovoljavajuće proradi, aplikacija se uprži u ROM.

PSOS podržava 32 nivoa prioriteta koja mogu biti dodijeljena tasku. U minimalnoj konfiguraciji, otisak ciljnog operativnog sistema je samo 12 Kbajta. Za dijeljenje kritičnih resursa između real-time taskova, PSOS podržava nasljeđivanje prioriteta i sinhronizacijski protokol (priority ceiling protocol). Podržava segmentirano upravljanje memorijom i ne podržava virtualnu memoriju obzirom da je namjenjen za korištenje u malim i srednjim ugrađenim aplikacijama.

UNIX I WONDOWS KAO RTOS

PSOS definiše memorijski region (memory region) koji je fizički kontinualan blok memorije. Memorijski region je kreiran od strane operativnog sistema kako bi se mogao pozvati od strane aplikacije. Programer može dodjeliti (alocirati) task memorijskom regionu.

Kod većine savremenih operativnih sistema, kontrola se prebacuje na kernel kada se desi interapt. PSOS koristi drugačiji pristup. Drajveri uređaja su izvan kernela i mogu biti učitani i uklonjeni za vrijeme run time-a. Kada se dogodi interapt, procesor skače direktno do servisne rutine prekida na koju pokazuje tabela vektora. Namjera nije samo ubrzanje već i da se programeru da puna kontrola nad rukovanjem izuzecima.

.

UNIX I WONDOWS KAO RTOS

VRTX

VRTX (**Versatile Real-Time Executive**) je operativni sistem proizveden od strane Mentor Graphics. Podržava POSIX-RT standard. VRTX je certificiran od strane US FAA (Federal Aviation Agency) i koristi se za misiju u kritičnim aplikacijama kao što je avionika. VRTX je dostupan u dva multitasking kernela: VRTXsa (scalable architecture) i VRTXmc (micro controller).

VRTXsa se koristi za velike i srednje aplikacije. Podržava virtualnu memoriju. To je POSIX spremna biblioteka i podržava nasljeđivanje prioriteta. Njegovi sistemski pozivi se obavljaju deterministički u fiksnim vremenskim intervalima i u potpunosti je priemptibilan. VRTXmc je optimiziran za potrošnju energije i veličine ROM-a i RAM-a. Također, ima veoma mali memorijski otisak. Kernel obično zahtijeva 4 do 8Kbajta ROM-a i 1Kbajt RAM-a. Ne podržava virtualnu memoriju. Ova verzija je namjenjena za upotrebu u ugrađenim aplikacijama kao što su igračke, mobiteli i drugi ručni uređaji (SoC- system on the chip).

UNIX I WONDOWS KAO RTOS

VxWorks

VxWorks je proizvod firme Wind River Systems. To je host-target vrsta real-time operativnog sistema. Host može biti mašina bazirana na Windows ili Unix operativnom sistemu. VxWorks je u skladu sa POSIX-RT standardom. Dolazi sa integrisanim razvojnim okruženjem zvanim Tornado. Kao dodatak standardnoj podršci za razvoj aplikacije kao što je editor, cross-kompajler, cross-debugger, itd. Tornado sadrži VxSim i WindView. VxSim simulira VxWorks target za prototipisanje i testiranje okruženja u nedostatku prave ciljne ploče. WindView sadrži alate za debuggiranje za okruženje simulatora. VxMP je višeprocesorska verzija VxWorksa.

VxWorks je iskorišten u Mars Pathfinderu koji je poslat na Mars 1997 godine. Pathfinder se spustio na Mars, odgovarao je na komande sa Zemlje, i počeo da šalje naučne i inženjerske podatke.

UNIX I WONDOWS KAO RTOS

Međutim, problem je bio da se Pathfinder povremeno resetovao. Inženjeri su udaljeno, korištenjem VxWorks alata za debugging ustanovili da je problem uzrokovala inverzija prioriteta što je uzrokovalo da real-time taskovi promaše zadati rok što je rezultovalo da procedura za rukovanje izuzecima svaki put resetuje sistem. Iako VxWorks podržava nasljeđivanje prioriteta, utvrđeno je, korištenjem alata za debuggiranje, da je bio isključen u konfiguracijskom fajlu. Problem je riješen njegovim ponovnim omogućavanjem.

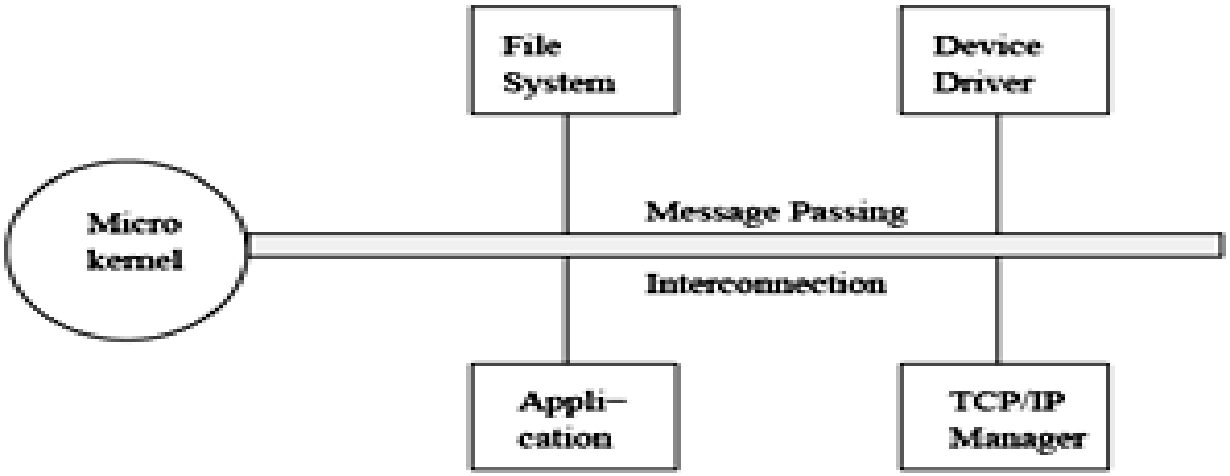
UNIX I WONDOWS KAO RTOS

QNX

QNX je proizvod firme QNX Software System Ltd. QNX je namjenjen za upotrebu u mission-citical operacijama u područjima kao što su medicinski instrumenti, internet ruteri, telemetrijski uređaji, kontrola procesa, zračna kontrola saobračaja. QNX Neutrino nudi POSIX API i implementiran je korištenjem mikrokernel arhitekture. QNX Mikrokernel arhitektura je prikazana na narednoj slici. Zbog fino granulirane skalabilnosti mikrokernel arhitekture, može biti konfigurisan na veoma malu veličinu – kritična prednost u uređajima kod kojih čak i 1% redukcije memorije može rezultirati milionima dolara profita.

Neutrino i njegov „microGUI“ – nazvan Photon, je dizajniran da radi ekstremno brzo sa veoma malim zauzećem memorije, što čini njegovo uključivanje u portabilne uređaje mogućim.

UNIX I WONDOWS KAO RTOS



QNX mikrokernel arhitektura

Zapravo, QNX i Neutrino već pokreću Web bazirane uredjaje, set-top boxove, MP3 playere, opremu korištenu u industriji i medicini. QNX Neutrino je prenesen na veliki broj platformi i danas se pokreće na većini savremenih procesora koji se koriste na tržištu ugrađenih uređaja. Ovo uključuje Intel x86 familiju, MIPS, PowerPc i ARM familiju procesora.

UNIX I WONDOWS KAO RTOS

µC/OS-II

µC/OS-II proizvodi Micrium Corporation. Ovaj real-time operativni sistem je napisan u ANSI C-u i sadrži male dijelove asemblerskog koda. Dijelovi asemblerskog koda su minimalni kako bi se system mogao lahko prebaciti na različite procesore. Do danas, ovaj operativni sistem je prebačen na preko 100 različitih procesorskih arhitektura počevši od 8-bitnih do 64-bitnih mikroprocesora, mikrokontrolera, itd. Neke bitne osobine operativnog sistema µC/OS-II su:

- µC/OS-II je dizajniran da dopusti programerima da imaju na raspolaganju za korištenje samo nekoliko od ponuđenih servisa ili da odaberu kompletnu gamu servisa. Ovo dopušta programerima da minimiziraju količinu memorije neophodnu za ovaj operativni sistem.

UNIX I WONDOWS KAO RTOS

- µC/OS-II ima u potpunosti priemptivan kernel. Ovo znači da µC/OS-II uvijek obezbjeđuje da će task najvećeg prioriteta koji je spreman biti uzet na izvršavanje.
- Dopušta kreiranje do 64 taska. Svaki task zahtijeva da radi na jedinstvenom nivou prioriteta, između 64 nivoa prioriteta. Round-robin raspoređivanje nije podržano. Nivoi prioriteta se koriste kao PID (Proces Identifier) za taskove.
- µC/OS-II koristi particionisanu memorijsku shemu. Svaki dio memorije se sastoji od nekoliko blokova fiksne veličine. Alokacija i dealokacija memorijskih blokova fiksne velicine se vrši u konstantnom vremenu i deterministička je. Task može kreirati i koristiti više memorijskih particija, tako da se mogu koristiti memorijski blokovi različitih velićina.
- µC/OS-II je certificiran od strane FAA (Federal Aviation Administration) za komercijalnu primjenu u avijaciji.

UNIX I WONDOWS KAO RTOS

RT Linux

Linux je besplatni operativni sistem. To je robustan, mogućnostima bogat i efikasan operativni sistem. Međutim, sam po sebi Linux je operativni sistem opšte namjene. Dostupno je nekoliko real-time implementacija Linux-a (RTLinux). Razmotrit ćemo neke od generičkih osobina real-time Linux sistema. RTLinux je self-host operativni sistem. RTLinux se izvršava uporedo sa Linuxom. Real-time kernel stoji između hardvera i Linux sistema. Prema standardnom Linux kernelu, RTLinux sloj se pojavljuje kao hardver. RTLinux kernel presreće sve interapte generisane od strane hardvera. Hardverski interapti koji nisu povezani sa real-time aktivnostima se proslijeđuju Linux kernelu kao softverski interapti kada je RTLinux kernel nezaposlen (idle) i izvršava se standardni Linux kernel. Ako interapt prouzrokuje da se pokrene real-time task, real-time kernel priemptira Linux, ako se Linux izvršava u tom momentu i pušta real-time task da se izvršava. Prema tome, Linux se

UNIX I WONDOWS KAO RTOS

izvršava kao RTLinux pozadinski task niskog prioriteta. U pristupu implementiranom kod RTLinuxa, postoje dva nezavisna kernela: real-time kernel i Linux kernel. Ovakav pristup je također poznat kao *dual kernel* pristup. Pošto je real-time kernel implementiran izvan Linux kernela, bilo koji task koji zahtijeva deterministično raspoređivanje izvršava se kao real-time task.

U poređenju sa mikro-kernel pristupom, slijedi nekoliko bitnih nedostataka dual-kernel pristupa.

- Dvostruki napori pri kodiranju
- Krhko okruženje
- Ograničena portabilnost
- Poteškoće kod programiranja

UNIX I WONDOWS KAO RTOS

Lynx

Lynx je self-host real-time operativni sistem. To je operativni sistem baziran na mikro-kernelu. Lynx je u potpunosti kompatibilan sa Linuxom. Sa Lynxovom binarnom kompatibilnošću, Linux program (binary image) može biti direktno pokrenut na Lynxu. S druge strane, za ostale Linux kompatibilne operativne sisteme kao što je QNX, Linux aplikacije treba da budu najprije rekompajlirane kako bi se mogle izvršiti na njima. Mikrokernel Lynxa je veličine 28 KB i obezbjeđuje najbitnije servise za raspoređivanje taska, rad sa interaptima, i sinhronizaciju. Ostali servisi su obezbjeđeni kao kernelovi plug-inovi (KPI). Dodavanjem ovih plug-inova mikro-kernelu, sistem se može konfigurisati da podrži I/O, fajl sisteme, itd. Sa punom konfiguracijom, može funkcionisati kao Unix mašina na kojoj se mogu izvršavati mehki i tvrdi real-time taskovi. Za razliku od mnogih ugrađenih operativnih sistem Lynx podržava zaštitu memorije.

UNIX I WONDOWS KAO RTOS

Windows CE

Windows CE je ogoljena verzija Windows operativnog sistema. Ima minimalno zauzeće memorije od 400 KB. Omogućava 256 nivoa prioriteta. Da bi optimizirao performanse, sve niti se izvršavaju u kernel modu. Preciznost tajmera je 1ms za sleep i wait API. Različite funkcionalnosti kernela su razbijene na male nepriemptivne sekcije. Kao rezultat toga, za vrijeme sistemskog poziva priempcija taska se isključuje na kratke periode vremena. Također, Windows CE podržava ugnježdene interapte. Koristi jedinicu za upravljanje memorijom (MMU) za upravljanje virtualnom memorijom. Windows CE koristi shemu nasljeđivanja prioriteta kako bi izbjegao problem inverzije prioriteta koji je prisutan kod Windows NT operativnog sistema.