SUVREMENI OPERACIJSKI SUSTAVI I UPRAVLJANJE PROCESIMA

Dejanović, Jelena

Sveučilište Josipa Jurja Strossmayera u Osijeku
Ekonomski fakultet u Osijeku
Preddiplomski studij (Poslovna informatika)

Jelena Dejanović

SUVREMENI OPERACIJSKI SUSTAVI I UPRAVLJANJE PROCESIMA

Završni rad

Osijek, 2019.

Sveučilište Josipa Jurja Strossmayera u Osijeku
Ekonomski fakultet u Osijeku
Preddiplomski studij (Poslovna informatika)

Jelena Dejanović

SUVREMENI OPERACIJSKI SUSTAVI I UPRAVLJANJE PROCESIMA

Završni rad

Kolegij: Informatika
JMBAG: 00102186365
e-mail: jdejanovic@efos.hr

Mentor: (prof. dr. sc. Josip Mesarić)

Osijek, 2019.

Josip Juraj Strossmayer University of Osijek
Faculty of Economics in Osijek
Undergraduate Study (Business Informatics)

Jelena Dejanović

MODERN OPERATING SYSTEMS AND PROCESS MANAGEMENT

Final paper

Osijek, 2019.

IZJAVA

O AKADEMSKOJ ČESTITOSTI,
PRAVU PRIJENOSA INTELEKTUALNOG VLASNIŠTVA,
SUGLASNOSTI ZA OBJAVU U INSTITUCIJSKIM REPOZITORIJIMA
I ISTOVJETNOSTI DIGITALNE I TISKANE VERZIJE RADA

1. Kojom izjavljujem i svojim potpisom potvrđujem da je završni rad isključivo rezultat osobnoga rada koji se temelji na mojim istraživanjima i oslanja se na objavljenu literaturu. Potvrđujem poštivanje nepovredivosti autorstva te točno citiranje radova drugih autora i referiranje na njih.
2. Kojom izjavljujem da je Ekonomski fakultet u Osijeku, bez naknade u vremenski i teritorijalno neograničenom opsegu, nositelj svih prava intelektualnoga vlasništva u odnosu na navedeni rad pod licencom Creative Commons Imenovanje – Nekomercijalno – Dijeli pod istim uvjetima 3.0 Hrvatska.
3. Kojom izjavljujem da sam suglasna da se trajno pohrani i objavi moj rad u institucijskom digitalnom repozitoriju Ekonomskoga fakulteta u Osijeku, repozitoriju Sveučilišta Josipa Jurja Strossmayera u Osijeku te javno dostupnom repozitoriju Nacionalne i sveučilišne knjižnice u Zagrebu (u skladu s odredbama Zakona o znanstvenoj djelatnosti i visokom obrazovanju, NN br. 123/03, 198/03, 105/04, 174/04, 02/07, 46/07, 45/09, 63/11, 94/13, 139/13, 101/14, 60/15).
4. Izjavljujem da sam autorica predanog rada i da je sadržaj predane elektroničke datoteke u potpunosti istovjetan sa dovršenom tiskanom verzijom rada predanom u svrhu obrane istog.

Ime i prezime studentice: Jelena Dejanović

JMBAG: 00102186365

OIB: 38829838453

e-mail za kontakt: jelenasmajl@gmail.com

Naziv studija: Preddiplomski studij

Naslov rada: Suvremeni operacijski sustavi i upravljanje procesima

Mentorica rada: prof.dr.sc. Josip Mesarić

U Osijeku, 23. rujan 2019 godine

Potpis Jelena Dejanović

Suvremeni operacijski sustavi i upravljanje procesima

SAŽETAK

Svrha operacijskog sustava je pružiti okruženje u kojem korisnik može izvršiti programe na pogodan i učinkovit način. Jedan od ključnih zadataka operacijskog sustava je upravljanje procesima u obradi i izvršenju računalnih programa. U radu će biti prikazan opis operacijskih sustava, njihova uloga, razvoj kroz povijest te poznati koncepti među kojima će rad biti baziran na procesima. Također, u radu će biti prikazan i objašnjen način izvođenja procesa u računalu pod nadzorom operacijskog sustava kao i problemi te moguće nepoželjne situacije koje mogu nastati. Jedan od problema vezan je za prebacivanjem procesora s jednog procesa na drugi. Objasnit će se i rješenja za nastale probleme. Također, biti će opisani pravci budućeg razvoja operacijskog sustava u kontekstu upravljanja računalnim procesima.

Ključne riječi: Operacijski sustav, proces, procesor

Modern operating systems and process management

The purpose of the operating system is to provide an environment where the user can execute programs in a convenient and efficient manner. One of the key tasks of the operating system is to manage the processes in processing and executing computer programs. The paper will describe the operating systems, their role, development throughout history, and familiar concepts among which the work will be based on processes. Also, the paper will show and explain how to perform the process on a computer under the control of the operating system, as well as problems and possible undesirable situations that may arise. One of the problems is switching processors from one process to another. Solutions to the problems will also be explained. Also, directions for future development of the operating system in the context of computer process management will be described.

SADRŽAJ

1. Uvod

Operacijski sustav je program koji upravlja sklopovljem računala. Također, djeluje kao posrednik između korisnika i sklopovlja računala. Svrha operacijskog sustava je pružiti okruženje u kojem korisnik može izvršiti programe na pogodan i učinkovit način. Operacijski sustav je softver koji upravlja računalnim sklopovljem. Sklopovlje mora osigurati odgovarajuće mehanizme za osiguranje ispravnog i neometanog rada računalnog sustava. Zbog toga što je operacijski sustav složen i velik, mora biti kreiran dio po dio.

Jedan od tih dijelova je upravljanje procesima. Računalni program ne čini ništa ukoliko se njegove instrukcije ne izvršavaju na procesoru. Proces je jedinica rada u sustavu. Operacijski sustav je zbirka procesa, a čine ih korisnički i sistemski procesi. Korisnički procesi su procesi koje izvršava korisnički kod ili ga pokreće određena aplikacija. Procesi koje izvršavaju programi operacijskog sustava nazivaju se sistemskim procesima. Sistemski procesi se inicijaliziraju sa operacijskim sustavom te su ključni procesi potrebni sustavu za njegovo funkcioniranje kao što je procesno upravljanje, upravljanje memorijom, upravljanje periferijom, itd.

Svakom procesu je dan adresni prostor. Adresi prostor procesa sadrži strojni jezični kod, podatkovni segment (sadrži globalne, statičke, vanjske i druge varijable) te segment stoga. PCB je povezan sa svakim procesom te sadrži informacije kao što su ID procesa, set registara, brojač programa, stanje procesa (pokrenut, na čekanju,..) te ostale informacije potrebne za izvršenje procesa. PCB (procesni kontrolni blok) je podatkovna struktura koju održava operacijski sustav za svaki proces kako bi što efikasnije upravljao procesima. Proces traži razne sistemske resurse (procesor, memorija, ulazno/izlazni uređaji, datoteke, ...) kako bi izvršio zadani zadatak. Također, postoji mnogo procesa koji se izvršavaju u isto vrijeme.

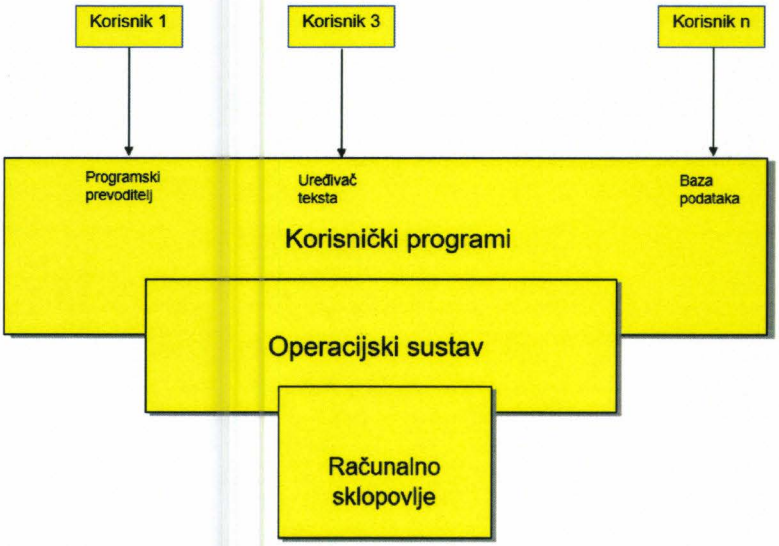
2. Teorijska podloga i prethodna istraživanja

2.1 Operacijski sustavi

Budin (2010) smatra da se računalni sustav može podijeliti na četiri komponente: korisnik, program, operacijski sustav te sklopovlje. Sklopovlje je opipljivi dio računala, a sastoji se od procesora, memorije te različitih ulazno-izlaznih uređaja. Sklopovlje neće funkcionirati bez određene programske podrške koja stroj i komponente vidi kao cjelinu. Programi služe za rješavanje različitih problema korisnika kao što su programski prevoditelji, uređivači teksta, sustav za upravljanje bazom podataka i tako dalje. Korisnici su ljudi, strojevi i druga računala koji nastoje uporabom računalnog sklopovlja riješiti svoje različite probleme. Prema Caru (2015), operacijski sustav je skup programa koji upravljaju izvršavanjem aplikacijskih programa i djeluju kao posrednik između korisnika računala i računalnog sklopovlja Omogućuje korisnicima i programima jednostavno korištenje sredstava sustava kroz odgovarajuća sučelja. Korisnici i programi stoga ne moraju poznavati (složene) detalje sklopovlja koje se koristi radi izvedbe njihovih naredbi.

2.1.1 Uloga operacijskog sustava

Slika 1. prikazuje povezanost korisnika, korisničkih programa, operacijskog sustava i računalnog sklopovlja.



Slika 1. Komponente računalnog sustava

Izvor: (Silberschatz, 2012, str. 4.)

Operacijski sustav može se gledati sa dva stajališta: korisnički i sklopovski pogled.

"Korisnički pogled računala odnosi se na sučelje koje se koristi. Takvi su sustavi dizajnirani da jedan korisnik monopolizira svoje resurse kako bi maksimalizirao posao koji se obavlja." (Silberschatz, 2012:5). Cilj je djelotvorno iskorištavanje raspoloživih strojnih komponenti i programa. U tim je slučajevima operacijski sustav dizajniran uglavnom za jednostavniju uporabu, s određenom pažnjom koja se posvećuje performansama, a nijedna se ne posvećuje korištenju resursa. U drugim slučajevima, moguće je korištenje resursa računala od strane više korisnika. Ti korisnici dijele resurse i razmjenjuju informacije

Sklopovski pogled odnosi se na operacijski sustav kao dodjelitelj resursa. Operacijski sustav dodjeljuje i upravlja računalnim resursima potrebnim za rješavanje specifičnih problema kao što su procesorsko vrijeme, memorijski prostor, prostor na disku za zapis datoteka i ulazno-izlazni uređaji na efikasan način, bez konflikata, dovoljno brzo i na zadovoljstvo svih korisnika.

Prema Budinu (2010) zadaci operacijskog sustava su:

- upravljanje procesima,
- upravljanje radnom memorijom,
- upravljanje sekundarnom memorijom (vanjskim memorijama),
- upravljanje ulazom/izlazom,
- upravljanje datotekama,
- zaštita dijelova sustava,
- otkrivanje pogrešaka u radu sustava,
- tumačenje upravljačkih naredbi,
- upravljanje mrežom računala.

2.1.2 Povijest operacijskih sustava

Operacijski sustavi usko su vezani za razvoj računala na kojima se pokreću te se prema tome i oni razvijaju. Svaka generacija razvoja računala postavlja nove zahtjeve prema sistemskom softveru te tako određuje razvoj operacijskih sustava.

1. Generacija (1945-55): računala sa elektronskim cijevima i prekidačima

Prvu generaciju obilježilo je prvo elektroničko računalo Colossus koje se sastojalo od oko dvije tisuće elektronskih cijevi, svrha mu je bila dekriptiranje njemačkih poruka. U ratne svrhe razvijeno je računalo sa 18 000 elektronskih cijevi pod nazivom Eniac. Eniac se smatra prekretnicom povijesti računala kakva se poznaje danas. (Povijest računala, https://racunalapc.weebly.com/).

2. Generacija (1955 – 1965): tranzistori i serijska obrada

Prema Tanenbaumu (1993), izumom tranzistora sredinom 1950-tih dolazi do velikog računalnog razvoja. Računala postaju pouzdana za proizvodnju i prodaju te postaju dugotrajnija. Korištena su za znanstvene svrhe te za matematičke proračune. Uobičajeni operacijski sustav je bio FSM (Fortran Monitor System) te IBSYS (IBM-ov operacijski sustav za 7094).

3. Generacija (1965 – 1980): Integrirani krugovi i višeprogramski sustavi

Prema Tanenbaumu (1997), u 3. generaciji dolazi do zamaha razvoja računala za širu upotrebu razvojem operacijskog sustava OS/360. OS/360 uvodi novu tehniku koja se zove multiprograming . U dosadašnjoj obradi, kada je određeni posao u stanju pauze ili u stanju čekanja da se U/I operacija završi, procesor se nalazi u stanju mirovanja. Višeprogramiranjem se navedeni problem rješava dijeljenjem memorije u više dijelova, na takav način da je svaki program dodijeljen drugom dijelu podijeljene memorije. Dok određeni program čeka završetak U/I operacije, drugi program je dodijeljen procesoru. U višekorisničkom radu uvedena je tehnika preraspodjele procesora prema aktivnosti korisnika, dodjeljujući češće procesor aktivnijim korisnicima koja se zove *timesharing*.

4. Generacija (1980 – sadašnjost): Osobna računala

Prema Tanenbaumu (2015), razvojem LSI (*Large-scale Integration*) krugova konstrukcijom čipova i smještanjem više tisuća tranzistora na silicijsku pločicu, dolazi doba razvoja osobnih

računala. LSI krugovi su integrirani krugovi koji sadrže manje od 100 000 tranzistora. Osobna računala postaju znatno jeftinija i pogodnija za širu upotrebu individualnih osoba.

Dolazi do razvoja OS za osobna računala. Jedan od prvih je CP/M operacijski sustav kojeg su koristili procesori Intel 8080, Zilog Z80 te mnogi drugi procesori zbog njegove dominacije u svijetu osobnih računala. Konstruiranjem IBM PC-a dolazi do razvoja MS-DOS operacijskog sustava, koji se kasnije koristio i na 80386 te 80486 procesorima te sadržavao i mnoge napredne značajke preuzete od UNIX-a u kasnijim izvedbama. Prema Jelenkoviću (2018), razvojem grafičkog sučelja u Mac OS X operacijskog sustava u Macintosh računalima dolazi i do razvoja Windows 3.1 te Windows 3.11 operacijskog sustava koji također podržavaju grafičko sučelje. Kasnije se razvijaju Windows 95, Windows 98, Windows NT, Windows NT 4.0, Windows 2000, Windows Me, Windows XP, Windows Server 2003, Windows Vista, Windows 7, Windows 8 te Windows 10. (Baksa, 2004)

Razvoj operacijskih sustava je i dalje uvjetovan razvojem računalne tehnologije i prilagođavat će se zahtjevima krajnjih korisnika.

2.1.3 Vrste operacijskih sustava

Razvoj operacijskih sustava utjecao je na različitost strukture i namjene operacijskih sustava. Operacijski sustavi se koriste u različitim okruženjima te s obzirom na to postoje:

- Mainframe operacijski sustavi: operacijski sustavi za centralizirana središnja računala koja koriste povezani korisnici sa različitih mjesta. Uglavnom su to velike organizacije koje te sustave koriste za rješavanje zahtjevnih i složenih obrada podataka. Primjeri su OS/360 i OS/390 te Linux.

- Server operacijski sustavi: operacijski sustavi koji pružaju usluge više korisnika odjednom preko mreže i omogućuju dijeljenje sklopovskih i softverskih resursa. Primjer su Unix, Linux Server i Windows Server.

- Višeprocesorski operacijski sustavi: Višeprocesorski sustav sastoji se od nekoliko procesora koji imaju zajedničku fizičku memoriju. Višeprocesorski sustav omogućuje veću računalnu snagu i brzinu. U višeprocesorskom sustavu svi procesori rade pod

jednim operacijskim sustavom. Prednost je poboljšanje performanse, veća propusnost sustava radi izvođenja više zadataka istodobno od strane različitih procesora te brže izvršavanje pojedinih zadataka. Primjeri su Linux i Windows.

- Operacijski sustavi za osobna računala: glavni zadatak takvog sustava je pružanje dobre podrške jednom korisniku. Primjeri su WINDOWS 98, WINDOWS Millenium, WINDOWS 2000, LINUX, Macintosh OS, Windows XP, Windows 7, Windows 8, Windows 10, itd.

- Operacijski sustavi u stvarnom vremenu: Karakteristika tih sustava se ogleda u tome da im je vrijeme ključan parametar. Dijeli se na: Hard real time i soft real – time.
  Hard real – time sustav se koristi dosta često u industrijskim procesima gdje računala moraju u stvarnom vremenu skupiti podatke u vezi produkcijskih procesa te na osnovu njih vršiti kontrolu stroja u tvornici. Ako se određena akcija ne dogodi u danom trenutku u kojem je zadano da se dogodi može dovesti štete u proizvodnoj liniji.
  Soft real – time sustavi su oni sustavi gdje ukoliko se propusti određeni rok za odgovor sustava je prihvatljivo, iako neželjeno, neće napraviti bilo koju trajnu štetu. Primjer ovog sustava su razni digitalni zvučni i multimedijalni sustavi, kao i današnji "pametni" telefoni.

- Handheld operacijski sustavi:

  Sadržana su na svim računalima malih dimenzija kao što su tableti i "pametni" telefoni. Najpoznatiji su Google-ov Android operacijski sustav te Apple-ov iOS. Većinu uređaja pogoni višejezgreni procesor.

- Namjenski operacijski sustavi:
  "Namjenski operacijski sustav je sustav koji ne dozvoljava korisničke instalacijske programe i sadržani su na upravljačkim uređajima koji se ne smatraju računalima u pravom smislu riječi." (*Tanenbaum, 2015:36*). Uobičajeni primjer su: mikrovalna pećnica, TV, automobil, DVD snimači, telefoni starijih generacija, MP3 player-i, itd. Ono što ih razlikuje od Handledheld operacijskih sustava je u tome što se na namjenskim sustavima nikada ne može pokrenuti softver sumnjivog porijekla. Sav program im je sadržan u trajnoj memoriji što dovodi do pojednostavljenja dizajna jer

nije potrebna zaštita između aplikacija. Najpoznatiji operacijski sustavi u ovoj domeni su Embedded Linux, QNX i VxWorks.

- Senzorski operacijski sustavi:
  "Senzorski operacijski sustavi se koriste u malenim računalima koja komuniciraju jedna s drugima te sa baznom stanicom koristeći se bežičnom komunikacijom." (*Tanenbaum, 2015:37*). Senzorska mreža se koristi za zaštitu okoline građevina, čuvanja nacionalnih granica detekciji požara u šumama, mjerenju temperature, predviđanje vremena, itd. Uređaji koji pogone ove sustave moraju biti dizajnirani da rade duži vremenski period u težim vremenskim prilikama te sadrže procesor, radnu i trajnu memoriju te jedan ili više senzor za praćenje okoline. Senzorski operacijski sustavi moraju biti maleni i jednostavni budući da uređaji koji ih pogone imaju vrlo malo radne memorije. Također, veliki problem je i životni vijek baterije. Kao i namjenski sustavi, programi se učitaju unaprijed te korisnik ne može učitati program naknadno. Poznati primjer senzorskog operacijskog sustava je TinyOS.

- Operacijski sustav pametnih kartica:
  Operacijski sustavi pametnih kartica se pokreću na uređajima reda veličine kreditnih kartica koji sadrže procesorski čip. Uređaji imaju ograničenu procesorsku snagu i memoriju. Mnoge kratice imaju jednu svrhu kao što je elektroničko plaćanje. Neke pametne kartice mogu biti Java orijentirane, što znači da trajna memorija može sadržavati prevoditelj JVM (Java Virtual Machine) te vršiti svrhu višeprogramiranja ukoliko se više manjih programa učita na karticu te prevedu pomoću JVM prevoditelja.

2.1.4 Koncepti operacijskog sustava

U koncepte operacijskog sustava svrstavaju se:

- procesi
- zastoji
- upravljanje memorijom

- rad sa ulazno-izlaznim uređajima
- datotečni sustav
- sigurnost
- tumač naredbi
- sistemski poziv

2.2. Procesi

Proces je osnovni koncept operacijskog sustava. Hansen (2001) smatra da je proces program u izvršenju. Proces nije isto što i programski kod, već puno više. To je aktivna cjelina za razliku od programa koji se smatra pasivnom cjelinom. Procesna svojstva uključuju stanje sklopovlja, memoriju, procesor i tako dalje.

Procesna memorija je podijeljena u četiri dijela za učinkovit rad:

- Odjeljak tekst (eng. Text section) sastoji se od prevedenog (kompiliranog) programskog koda, očitanog iz neizbrisivog prostora za pohranu kada se program pokrene.
- Odjeljak podaci (eng. Data section) sačinjavaju globalne i statičke varijable dodijeljene i inicijalizirane prije izvršenja glavne funkcije.
- Odjeljak hrpe (eng. Heap) se koristi za dinamičku raspodjelu memorije i upravljana je putem poziva „new", „delete", „malloc", „free",..
- Odjeljak stoga (eng. Stack) koristi se za lokalne varijable. Prostor na stogu je rezerviran za lokalne varijable kada su deklarirane. (Izvor: https://www.studytonight.com/operating-system/operating-system-processes, [pristupljeno 12. rujna. 2019]).

2.2.1 Modeli procesa

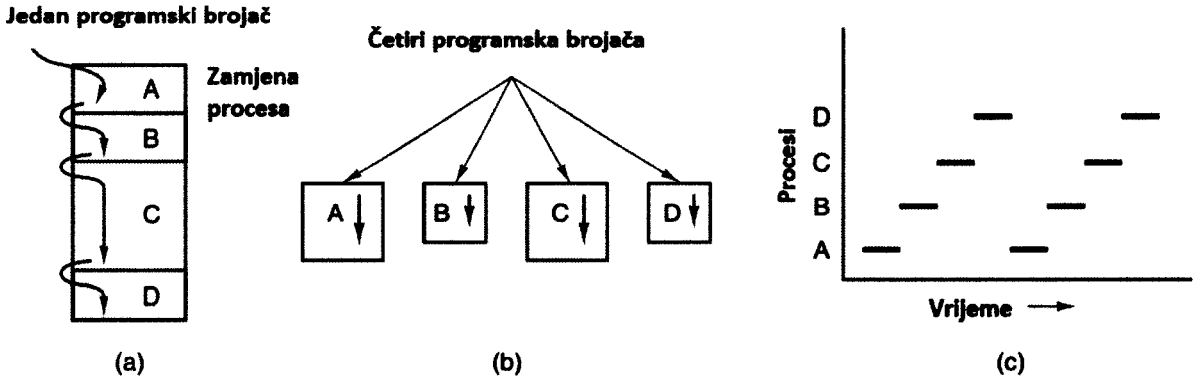
Proces je samo instanca izvršnog programa, a uključuje trenutne vrijednosti programskog brojača, registara i varijabli. Konceptualno, svaki proces ima svoj virtualni procesor. U

stvarnosti procesor se prebacuje sa procesa na proces. To ubrzano prebacivanje nazivamo multiprograming.

Slika a) prikazuje računalno višeprogramiranje za četiri programa u memoriji.

Slika b) prikazuje četiri procesa sa vlastitim kontrolnim tokom (vlastiti logički programski brojač) i svaki od procesa se izvršava neovisno jedan o drugom.

Slika c) prikazuje vremenski interval četiri procesa u kojemu je svaki od procesa ostvario određeni napredak, ali u jednom trenutku u stvarnosti samo jedan proces se izvršava.



Slika 2. Multiprograming

Izvor: (Kovačić, 2008, str. 12.)

Proces ima program, ulaz, izlaz i stanje. Jedan procesor može biti dijeljen između više procesa s planiranim algoritmima za odlučivanje kada prestati procesuirati određeni proces, a kada započeti procesuirati drugi proces. Za razliku, program je nešto što se može skladištiti na tvrdi disk i ne činiti ništa.

Važno je napomenuti da ako je program pokrenut dva puta da se računa kao dva procesa.

2.2.2 Procesno planiranje

Procesno planiranje je čin određivanja koji je proces u stanju pripravnosti i treba ga premjestiti u pokretno stanje. Prema Silberschatzu (2012) glavni cilj sustava za planiranje procesa je držati procesor cijelo vrijeme zauzetim i osigurati minimalno vrijeme odziva za sve programe. Da bi se to postiglo, procesni planer mora primijeniti odgovarajuća pravila za zamjenu procesa ulaza i izlaza procesora.

Procesni planer pripada jednoj od dvije opće kategorije:

- Planiranje koje nije preventivno - kada trenutačno izvršavani proces dobrovoljno odustane od resursa procesora
- Preventivno planiranje – kada operacijski sustav odluči favorizirati drugi proces isključivanje trenutno izvršavanog procesa

Redovi planiranja:

- Svi procesi po ulasku u sustav pohranjuju se u red poslova
- Procesi u stanju pripravnosti postavljaju se u red čekanja
- Procesi u čekanju od dostupnosti uređaja smještaju se u red uređaja (postoje jedinstveni redovi uređaja na raspolaganju za svaki U/I uređaj).

Hrgarek (2006) smatra da se novi proces u početku postavlja u red pripravnosti. Ondje se nalazi sve dok ne bude odabran za izvršenje.

Nakon što se proces dodijeli procesoru i izvršava može se dogoditi jedan od sljedećih događaja:

- Proces može dati U/I zahtjev i staviti ga u U/I red.
- Proces može stvoriti novi podproces i čekat njegovo okončanje
- Proces može biti prisilno uklonjen iz procesora kao rezultat prekida i vraćen u red pripravnosti

Prema Silberschatzu (2012) postoje tri vrste planera, a to su dugoročni planer, kratkoročni planer i srednjoročni planer.

Dugoročni planer pokreće se rjeđe. On odlučuje koji program mora ući u red poslova. Iz redova poslova, program Job Processor odabire procese i stavlja ih u memoriju za izvršavanje. Primarni cilj planera poslova je održati dobar stupanj višeprogramiranja (istovremeno izvršavanje nekoliko različitih poslova). Optimalni stupanj višeprogramiranja znači da je prosječna brzina stvaranja procesa jednaka prosječnoj brzini odlaska procesa iz memorije izvršenja.

Kratkoročni planer se također naziva i procesorski planer te radi vrlo često. Glavni mu je cilj poboljšati performanse procesora i povećati brzinu izvođenja procesa. Ovaj planer uklanja procese iz memorije i na taj način smanjuje stupanj višeprogramiranja. Kasnije se proces može ponovno uvesti u memoriju i njegovo izvršavanje nastaviti tamo gdje je stalo. Ta shema se naziva zamjena (engl. *Swapping*).

Srednjoročni planer obavlja postupak zamjene. Zamjena može biti potrebna kako bi se poboljšala kombinacija procesa te zato što je promjena memorijskih potreba prekomjerno ispunila dostupnu memoriju pa zahtjeva oslobađanje memorije. Čitav proces je opisan u donjem dijagramu.

2.2.3 Stvaranje procesa

Operacijskim sustavima je potreban način za stvaranje procesa. Prema Ribariću (2011), u vrlo jednostavnim sustavima ili u sustavima dizajniranim za pokretanje samo jedne aplikacije, moguće je imati sve procese koji će biti prisutni. U nekim složenijim sustavima, potreban je način za stvaranje i zaustavljanje procesa prema potrebi. Izdvajaju se četiri stavke koje uzrokuju stvaranje procesa, a to su:

- inicijalizacija sustava
- poziv sistemskog poziva iz aplikativnog programa
- korisnikov zahtjev za stvaranje novog procesa
- pokretanje serijske (batch) obrade. (PadaKuu, 2017)

Pri pokretanju operacijskog sustava stvaraju se brojni procesi. Neki od njih su procesi prednjeg plana, odnosno procesi u kojima komuniciraju korisnici i na taj način obavljaju posao. Međutim, postoje i procesi koji su u pozadini i koji nisu povezani sa korisnicima, već

imaju određene funkcije. Procesi koji ostaju u pozadini za obavljanje određenih aktivnosti kao što su e-pošta, web stranice, vijesti, ispis, nazivaju se *demonima*. Veliki sustavi ih obično imaju na desetke.

2.2.4 Prestanak procesa

Nakon kreiranja procesa, on se pokreće i izvršava svoj posao. Nakon navedenog proces se zaustavlja, a mogući razlozi prema Tanenbaumu (2015) su:

- normalan završetak
- završetak uzrokovan fatalnom greškom (neovisno od procesa)
- završetak uzrokovan greškom (ovisno od procesa)
- uništen nekim drugim procesom (neovisno od procesa)

Većina procesa se završava jer su obavili svoj posao. Kada prevoditelj prevede program koji mu je dodijeljen, on izvršava sistemski poziv kojim navodi da je gotov. Internetski preglednici i slični programi uvijek imaju ikonu koju korisnik može odabrati i ukloniti sve otvorene datoteke. Drugi razlog završetka procesa je taj što proces može otkriti ogromnu pogrešku. Na primjer, ako korisnik upiše neku naredbu za sastavljanje programa i takve datoteke nema, prevoditelj najavljuje tu činjenicu i završava. Nakon toga se većinom pojavljuje poruka da korisnik pokuša ponovno. Treći razlog završetka je pogreška uzrokovana postupkom. Na primjer to je nedovoljna memorija ili dijeljenje sa nulom. Četvrti razlog mogućeg završavanja procesa je taj da proces izvršava sistemski poziv koji nalaže operacijskom procesu da ukine neki drugi postupak. U nekim sustavima kada se proces završi bilo dobrovoljno ili na neki drugi način, svi stvoreni procesi se ukidaju. Ni Unix ni Windows ne funkcioniraju na taj način.

2.2.5 Hijerarhije procesa

U nekim sustavima kada proces stvori drugi proces, nadproces i podproces, oni nastavljaju biti povezani na određen način. Podproces može također stvoriti nove procese kreirajući

procesnu hijerarhiju. U UNIX operacijskim sustavima proces i svi njegovi podprocesi i daljnji sljedbenici zajedno formiraju procesnu grupu. Kada korisnik šalje signal s tipkovnice, signal je dostavljen svim članovima procesne grupe trenutno povezane s tipkovnicom. Individualno, svaki proces može dohvatiti signal, ignorirati signal, napraviti zadanu radnju i biti uništen nekim drugim signalom.

U suprotnosti, Windows operacijski sustav nema koncept procesne hijerarhije. Svi procesi su jednaki. Jedina naznaka hijerarhije je kada je proces kreiran, nadprocesu je dana specijalna mogućnost rukovanja koja se može koristiti za kontrolu podprocesa. Mogućnost rukovanja podprocesima može se prenijeti na neke druge procese te samim tim poništiti hijerarhiju.

2.2.6 Stanja procesa

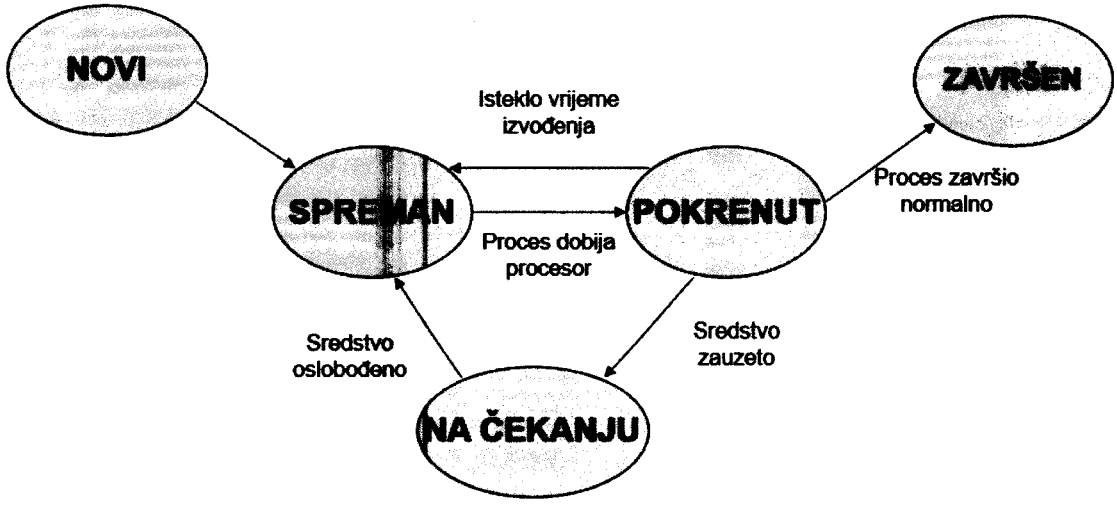
Kako se postupak izvršava, tako se mijenja i stanje procesa.

Procesi u operacijskom sustavu mogu biti u bilo kojem od sljedećih stanja:

- Novi (eng. New) - proces je kreiran
- Spreman (eng. Ready) – proces je u stanju čekanja da se dodijeli procesoru
- Pokrenut (eng. Running) – instrukcije se izvršavaju
- Na čekanju (eng. Waiting) – proces je u stanju čekanja da se dogodi neki događaj (prijem signala,..)
- Završen (eng. Terminated) – izvršenje procesa je završeno

Nazivi su proizvoljni i razlikuju se u različitim operacijskim sustavima, no ono što oni predstavljaju nalazi se na svim sustavima. Važno je znati da se na bilo kojem procesoru u bilo kojem trenutku može pokrenuti samo jedan proces.

Slika 3. predstavlja tranziciju stanja procesa. Kreiranjem novog procesa, proces prelazi u spremno stanje. Nakon što određeni proces dobije procesor on prelazi iz spremnog u pokrenuto stanje. Ukoliko je proces završio, on prelazi u završeno stanje. Ako je uslijed izvršenja procesa procesor krenuo izvršavati drugi proces, prvotni proces prelazi u stanje čekanja.



Slika 3. Tranzicija stanja procesa

Izvor: (Silberschatz, 2012, str. 108.)

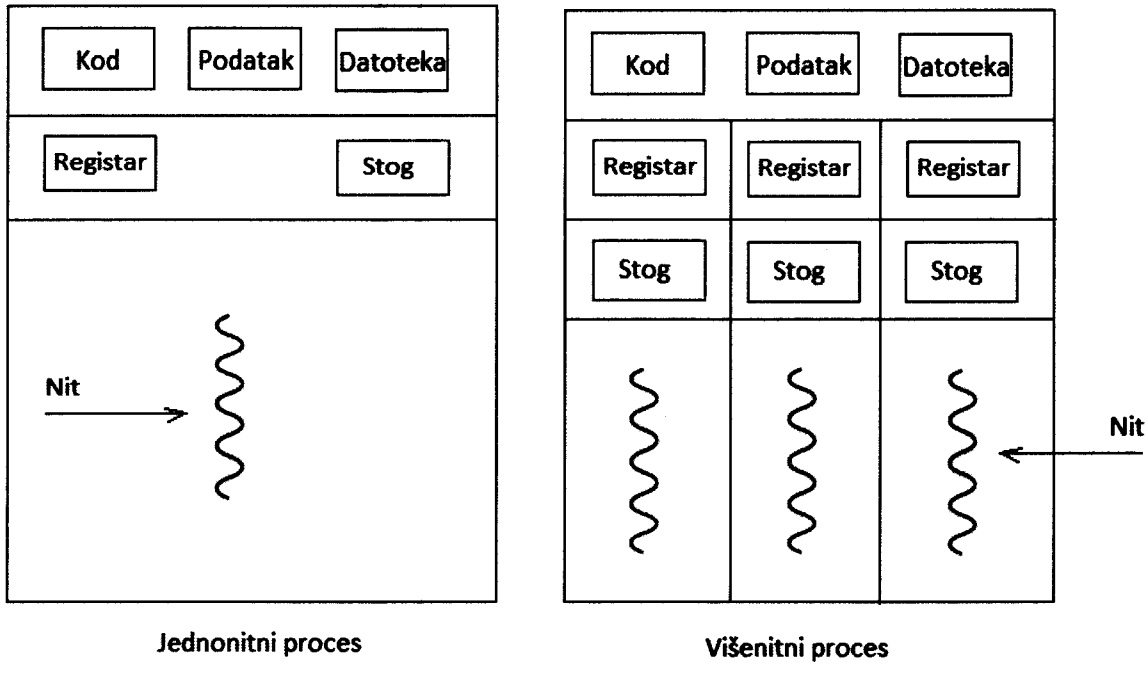
2.2.7 Blok upravljanja procesom (Proces Control Block)

Za svaki postupak postoji blok upravljanja procesa koji sadrži sve informacije o procesu. To je struktura podataka koja prema Silberschatzu (2012) sadrži sljedeće:

- Stanje procesa – može biti pokrenuto, na čekanju,..
- ID procesa i ID nadređenog procesa
- Registri procesora i programski brojač (programski brojač sadrži adresu sljedeće instrukcije koja će se izvršiti za taj proces)
- Informacije o rasporedu procesora – informacije o prioritetu i pokazivači na redove planiranja
- Informacije o upravljanju memorijom – tablice stranica i tablice segmenata
- Računovodstvene informacije – potrošeno vrijeme korisnika i jezgre procesora, brojevi računa, ograničenja,..
- Informacije o statusu U/I –dodijeljeni uređaji, otvaranje tablica datoteka,..7

2.2.8 Procesne niti

Pri izvođenju procesa izvodi se najmanje jedna procesna nit. Prema Silberschatzu (2012), nit je osnovna jedinica procesorske upotrebe. Sadrži identifikacijski broj, brojač, registarski set i stog. Dijeli s drugim nitima pripadnost istom procesu sa svojim programskim dijelom, podatkovnim dijelom i drugim resursima operacijskog sustava kao što su podaci i signali. Ako proces ima više niti za kontrolu, može izvršavati više od jednog zadatka u isto vrijeme.



Slika 4. Jednonitni i višenitni proces

Izvor: (Silberschatz, 2012, str.162.)

Slika 4. ilustrira razliku između jednonitnog procesa i višenitnog procesa.

Većina današnjih aplikacija pokretanih na modernim računalima su višenitne. Aplikacije su implementirane kao zasebni proces s nekoliko niti za kontrolu. Internet preglednik može sadržavati jednu nit za prikaz slika ili tekst, dok druga nit može dohvaćati podatke s mreže, treća za odgovor na pritisnutu tipku na tipkovnici, itd. Aplikacije također mogu biti dizajnirane da omoguće procesorske sposobnosti na višejezgrenim sustavima. One mogu izvršavati izrazito procesorske intenzivne zadatke u paraleli preko više jezgri procesora.

Na primjeru web servera koji prihvaća zahtjeve klijenata usmjerenih prema web stranici, slici, zvuku, itd. može se primijeniti proces s jednom niti. Ukoliko web server dobiva više zahtjeva

različitih korisnika za pristup te može dovesti u stanje zaposlenosti koje može rezultirati s podugim vremenom za servisiranjem zahtjeva klijenta. Razlog tome leži u činjenici da proces može servisirati samo jednog klijenta u isto vrijeme.

Neka od rješenja navedenog problema mogu se realizirati na način da pri primitku zahtjeva korisnika server stvara novi odvojen proces koji će servisirati traženi zahtjev. Navedena metoda stvaranja procesa je izrazito dugotrajna i resursno intenzivna, pogotovo jer više procesa odrađuju istovjetne zadatke. Prema Sterlingu (2018), je u pravilu efikasnije koristiti jedan proces koji sadrži više niti. U tom slučaju, server će stvoriti jedan proces unutar kojeg će stvoriti procesnu nit koja će dohvaćati zahtjeve klijenata. Kada dođe do zahtijeva, umjesto da se stvori novi proces, server će stvoriti novu nit koja će servisirat zahtjev te nastaviti stvarati nove niti za nove korisničke zahtjeve.

Prema Silberschatzu (2012), koristi multiprograminga i procesorskih niti:

- odzivnost – višenitna interaktivna primjena može dozvoliti programu da nastavi biti pokrenut čak i ako je djelomično blokiran ili izvršava dugotrajan zadatak, čime se povećava odzivnost za korisnika. Ova mogućnost je izrazito korisna u korisničkim sučeljima. Ukoliko korisnik pritisne tipku koja rezultira izvršavanjem izrazito vremenski zahtjevnog zadatka, program s jednom niti biti će nereagirajući za korisnika dok se zadatak ne izvrši. Ukoliko bi se zadatak izvršavao u odvojenoj niti, program bi ostao u reagirajućem stanju na zahtjev korisnika.

- djeljivost resursa – procesi mogu dijeliti resurse prema tehnikama zajedničke memorije i slanja poruka. Takve tehnike su izričito uređene od strane programera. Za razliku od procesa, niti dijele memoriju i resurse procesa kojem po običaju pripadaju. Prednost dijeljenja programskog koda i podataka je u tome što omogućuje primjenu nekoliko različitih niti unutar istog adresnog prostora.

- ekonomičnost – dodjela memorije i resursa u procesu stvaranja procesa je skup postupak. Budući da niti dijele resurse procesa kojem pripadaju, ekonomičnije je stvoriti niti.

- skalabilnost – prednosti višeprogramiranja mogu biti još veće u višeprocesorskim arhitekturama, gdje se niti mogu izvršavati u paraleli na različitim procesorskim

jezgrama. Proces s jednom niti može se izvršavati na jednom procesoru, bez obzira koliko ih je na raspolaganju.

3. Metodologija rada

3.1 Izvođenje procesa u računalu – zamjena procesa

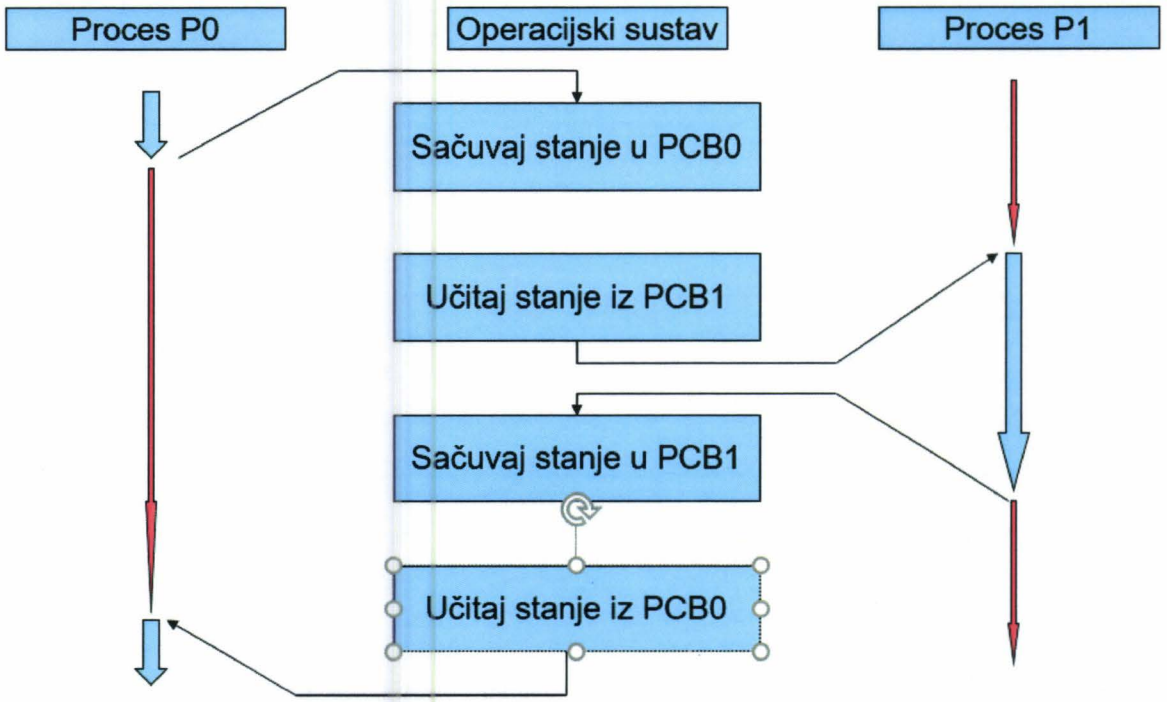
Zamjena procesa (engl. Context Switch) predstavlja zadatak prebacivanja procesora od jednog procesa prema drugom procesu. Čitav postupak zahtijeva spremanje stanja starog procesa i učitavanje spremljenog stanja za novi proces. Zamjena procesa predstavljena je blokom upravljanja procesom (PCB). Ona uključuje vrijednosti registra procesora, informacije o stanju procesa i informacije upravljanja memorijom. Kada dođe do zamjene procesa jezgra OS-a sprema informacije starog procesa u PCB i učitava spremljene informacije novog procesa koji se planira pokrenuti. Brzina zamjene procesa ovisi od stroja do stroja, o brzini memorije, broju registara koji se moraju kopirati i postojanju posebnih uputa. Brzine se kreću od 1 do 1000 μs. Iz razloga što sustav ne radi koristan posao tijekom zamjene procesa, kad god je moguće, programeri koriste nove strukture (niti), kako bi poboljšali performanse sustava.

Koraci zamjene procesa obuhvaćaju:

- Spremanje konteksta o procesu koji se trenutno odvija na procesoru
- Ažuriranje PCB-a
- Premještanje PCB-a prvotnog procesa u odgovarajuća stanja (stanje spremnosti, U/I stanja)
- Odabir novog procesa za izvršenje
- Ažuriranje PCB-a za odabrani proces (ažuriranje procesa na pokrenuto stanje)
- Ažuriranje strukture podataka za upravljanje memorijom prema potrebi
- Učitavanje prethodnih vrijednosti PCB-a i registara prethodno pokrenutog procesa

Kontekst zamjene procesa je "skupa operacija" zbog potpunog pražnjenja TLB (*translation lookaside buffer*) tablice (TLB Flush), dijeljenja priručne memorije (eng. *cache*), pokretanje planera zadataka,... TLB je priručna memorija koja se koristi kako bi se smanjilo vrijeme pristupa lokaciji korisničke memorije te sadrži ranije translacije iz virtualne prema fizičkoj memoriji. (Avgeriou, 2018)

Zamjena konteksta između dviju niti istog procesa je brže nego između dva procesa jer imaju iste virtualne mape. Zbog toga potpuno pražnjenje TLB tablice nije potrebno jer u većini slučajeva unosi će biti nevažeći. TLB tablica je malog kapaciteta, a svaki proces koristi različite dijelove memorije. Novi proces ne koristi mapiranja koja ostanu u TLB memoriji jer je on u nekom drugom opsegu memorijskih adresa.



Slika 5. Zamjena procesa u procesoru

Izvor: (Silberschatz, 2012, str. 109.)

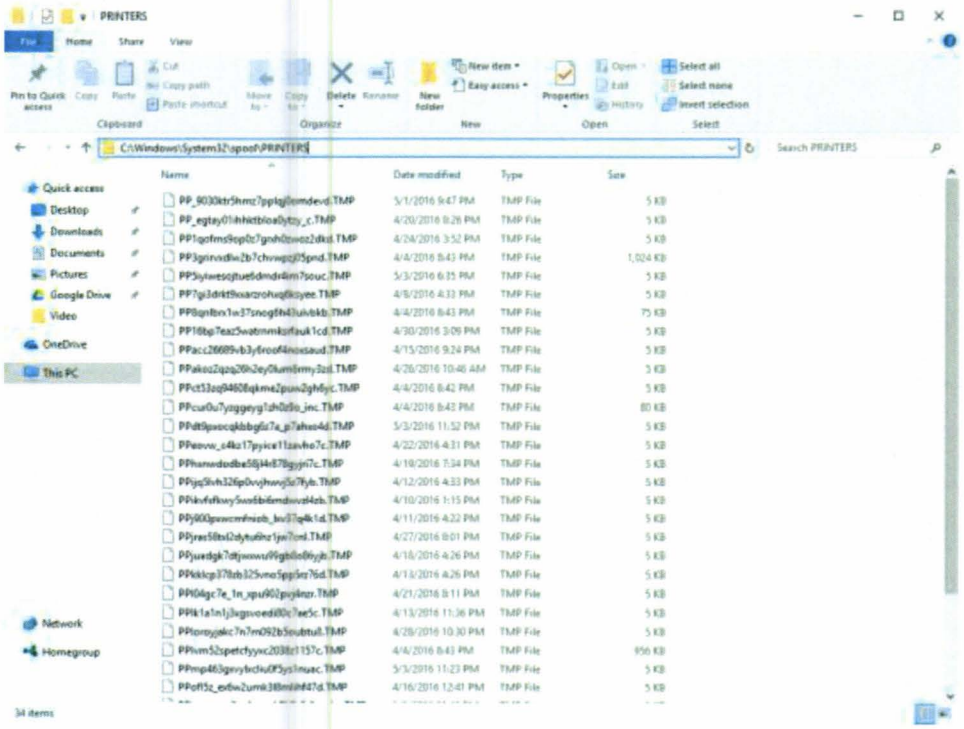
Slika prikazuje zamjenu procesa u procesoru. Proces P0 započinje obradu na procesoru. U jednom trenutku proces P0 gubi procesor, te ga dobiva proces P1. Proces P0 u tom trenutku čuva stanje u PCB-u. Proces P1 se izvršavao ranije te učitava stanje iz PCB-a te nastavlja s izvođenjem. Nakon nekog vremena proces P1 gubi procesor te radi zapis stanja na PCB. Proces P0 ponovno dobiva procesor te nastavlja s izvršavanjem učitavajući prethodno stanje iz PCB-a.

4. Opis istraživanja i rezultati istraživanja

Na sljedećim primjerima biti će prikazan problem vezan za zamjenu procesora između procesa. Primjenom zamjene procesora moguće su pojave grešaka prilikom izvršenju procesa. Na primjeru dokumenata za ispis na pisač biti će naveden primjer s mogućim poteškoćama u radu.

Direktorij u koji se smještaju dokumenti za ispis naziva se spooler direktorij, a proces odgovoran za preusmjeravanje dokumenta za ispis naziva se *printer daemon*. Proces je uvijek u stanju aktivnosti. Prilikom rada sa spooler direktorijem od važnosti su dvije varijable: ulazna i izlazna varijabla. Ulazna varijabla određuje slobodno mjesto za upis dokumenta u spooler direktorij, a izlazna odabire dokument za ispis.

Slika 6. prikazuje lokaciju *spooler* direktorija u kojemu se smještaju dokumenti spremni za ispis na pisač.



Slika 6. Spooler direktorij

Izvor: (thePC.co, 2019.)

4.1 Primjer 1. Izvođenje procesa u računalu

Za zadani primjer prikazane su sljedeće vrijednosti za ulazne i izlazne varijable: ulazna varijabla = 3, a izlazna varijabla = 1. Proces A je prvi zadani proces kojemu je dodijeljen procesor, a proces B je proces kojemu nije dodijeljen procesor.

Slijed izvođenja procesa:

- proces A dobiva procesor te provjerava slobodno mjesto u *spooler* direktoriju kako bi upisao novi dokument za ispis na pisač (ulazna varijabla iznosi 3).
- proces vrši upis dokumenta u *spooler* direktorij te povećava vrijednost ulazne varijable za jedan (ulazna varijabla iznosi 4).
- proces A gubi procesor, te je procesor dodijeljen procesu B.
- proces B vrši provjeru *spooler* direktorija te potvrđuje je li mjesto 4 slobodno
- proces B vrši upis drugog dokumenta u *spooler* direktorij te postavlja ulaznu varijablu na 5

4.2 Primjer 2. Izvođenje procesa u računalu

Za isti primjer naveden ranije prikazana je situacija koja dovodi do pojave greške prilikom ispisa na pisač.

Prikazane su sljedeće vrijednosti za ulazne i izlazne varijable: ulazna varijabla = 3, a izlazna varijabla = 1. Proces A je prvi zadani proces kojemu je dodijeljen procesor, a proces B je proces kojemu nije dodijeljen procesor.

Slijed izvođenja procesa:

- proces A dobiva procesor te provjerava slobodno mjesto u *spooler* direktoriju kako bi upisao novi dokument za ispis na pisač (ulazna varijabla iznosi 3),
- proces A registrira mjesto 3 kao slobodno mjesto te u tom trenutku gubi procesor (zbog toga što se proces A nije izvršavao dovoljno dugo, on nije povećao vrijednost ulazne varijable za jedan te ulazna varijabla i dalje iznosi 3),
- procesu B je dodijeljen procesor,

- proces B vrši provjeru *spooler* i provjerava je li mjesto 3 slobodno,
- proces B vrši upis drugog dokumenta u *spooler* direktorij na mjesto 3 te postavlja ulaznu varijablu na 4,
- proces B gubi procesor te ga dobiva proces A,
- proces A nastavlja izvođenje od mjesta prekida te kako je u prethodnoj provjeri *spooler* direktorija izvršio provjeru slobodnog mjesta, on ju ne ponavlja te sprema dokument za ispis na mjesto 3,
- proces A postavlja vrijednost ulazne varijable na 4.

4.3 Usporedba izvođenja procesa na promatranim primjerima

U prvom primjeru u opisu tijeka zapisa za ispis na pisač navedeni postupak zamjene procesora neće dovesti do pojave greške u izvođenju te će se oba dokumenta moći isprintati.

Drugi primjer pokazuje da iako su oba procesa izvršila sve ono što im je zadano da izvrše, u postupku dodjele procesora nastala je greška. Krajnji rezultat je da će samo jedan dokument izvršiti ispis na pisač i to će biti dokument zapisan od procesa A. Kako proces A nije stigao završiti zadano tijekom izvršenja na procesoru te je izgubio procesor nakon provjere slobodnog mjesta u *spooler* direktoriju, on nije stigao napraviti upis datoteke i povećati ulaznu varijablu za jedno mjesto. Proces B je napravio provjeru stanja te vrši upis u direktorij na mjesto koje je ranije trebao napraviti upis procesa A. Nakon što je proces A ponovno dobio procesor, on vrši upis datoteke na isto mjesto na kojemu je bio upis procesa B, što rezultira da će se samo jedan dokument ispisati.

5. Rasprava

U ranije navedenom primjeru gdje je prikazan problem vezan za ispis na pisač, moguće je određenim tehnikama smanjiti ili eliminirati mogućnost pojave pogreške u radu. Razlog tome leži u činjenici što je gubitak procesora moguć u bilo kojem trenutku izvođenja te je potrebno osigurati načine i metode za njegovo sprečavanje.

U situacijama natjecanja procesa (engl. *race condition*), ukoliko dođe do situacije kao gore navedena, gdje se dva procesa se natječu za rad nad jednim djeljivim resursom može se izvesti rješenje u fazi izvođenja procesa koja se naziva kritična sekcija (engl. *critical region*). To znači da ukoliko se jedan proces nalazi u kritičnoj sekciji, onemogućiti će pristup drugom procesu istom djeljivom resursu. U našem primjeru dogoditi će se da proces A dobivanjem procesora ulazi u kritičku sekciju te ju zadržava prilikom gubljenja procesora. Dobivanjem procesora proces B je onemogućen za pristup *spooler* direktoriju te neće vršiti provjeru i upis na mjesto označeno vrijednošću ulazne varijable. Nakon što proces A dobije procesor on će nastaviti od mjesta gdje je započeo te uspjeti završiti upis u spooler direktorij i uvećati vrijednost ulazne varijable za jedan. Navedena tehnika međusobnog isključivanja (engl. *mutual exclusion*) će omogućiti da ne dođe do greške u tumačenju slobodnog mjesta i kao rezultat omogućiti će ispis dokumenata na pisač. Problem navedene tehnike odražava se u tome što će proces B za cijelo vrijeme dodjele procesora provesti u fazi slanja upita za ulazak u kritičnu sekciju. Budući da proces A nije završio izvršavanje, proces B će cijelo vrijeme dobivati negativan odgovor, tj. biti će u fazi zaposlenog čekanja.

Navedena faza sprečava pojavu greška prilikom dodjela procesora procesima, ali donosi određen problem jer velik dio vremena procesor ne izvršava koristan rad (proces B slanjem zahtjeva za ulazak u kritičnu sekciju i dobivanjem istih negativnih odgovora). Jedan od načina da izbjegnemo nastalu situaciju je da onemogućimo prekide sve dok se proces nalazi u kritičnoj sekciji. To znači da proces koji započne kritičnu sekciju ne može izgubiti procesor sve dok ne završi izvođenje. Navedenim načinom omogućavamo da procesor izvršava koristan rad za cijeli vremenski period, ali dobivamo problem u vidu gubitka važnih signala stanja operacijskog sustava prilikom izvođenja kritične sekcije.

Drugačiji pristup može se ostvariti korištenjem varijabli kao kontrole ulaska u kritičnu sekciju. Ukoliko je vrijednost varijable 0, procesu je dozvoljen pristup za ulazak u kritičnu sekciju, a ukoliko je vrijednost varijable 1, ona označava da se određeni proces nalazi u

kritičnoj sekciji te neće dozvoliti pristup drugim procesima ulazak. Navedena tehnika radi na principu da ukoliko proces želi ući u kritičnu sekciju provjeriti će prvo vrijednost varijable. Ukoliko je 0, ulazi u kritičnu sekciju te mijenja vrijednost varijable u 1. Završetkom kritične sekcije vrača vrijednost varijable na 0 te omogućava drugim procesima ulazak u kritičnu sekciju. Ova tehnika može dovesti do pojave dva procesa u kritičnoj sekciji, a to će se dogoditi u situaciji kada proces A provjeri varijablu i prije nego što promijeni njen iznos iz 0 u 1 izgubi procesor. Vjerojatnost da se dogodi navedena situacije je puno manja nego u ranije navedenom primjeru vezanom za *spooler* direktorij.

Neka od rješenja mogu se izvesti sklopovskim putem pomoću TSL instrukcija. Rješenje je slično kao prilikom korištenja varijabli, ali sklopovskom implementacijom onemogućeno je da se dva procesa nađu u kritičnoj sekciji. TSL instrukcija kopira vrijednost varijable u registar prilikom ulaska procesa u kritičnu sekciju te ga postavlja na vrijednost 1. Ukoliko drugi proces dobije procesor on će prvo očitati vrijednost iz registra te vršiti zaposleno čekanje. Ova tehnika pomoću TSL instrukcija zahtjeva sklopovsku podršku te ju čini ovisnom o sklopovlju računala.

6. Zaključak

Tema rada bavi se problematikom procesa i dodijeljenog im procesora. U radu su predstavljeni problemi i moguće nepoželjne situacije koje mogu nastati prebacivanjem procesora sa jednog procesa na drugi. Također, predstavljena su određena rješenja za nastale probleme. Dana je usporedba načina rada procesora nad procesima te načina rada procesora nad nitima unutar jednog procesa.

Kao što je ranije u radu navedeno, višenitni rad ubrzava rad sustava jer radi u vlastitom virtualnom prostoru i ne zahtjeva zapis u PCB kao što to radi proces radi zapisa stanja obrade prilikom promjene procesa na procesoru. Višenitni rad je budućnost razvoja operacijskih sustava što analogno dovodi i do povećane složenosti u razvoju operacijskih sustava. Također, budući razvoj operacijskih sustava biti će vezan za razvoj tehnologije, ali će biti okrenut i prema budućim korisničkim zahtjevima suvremenog doba.

Literatura

1. Baksa, M., P&O (2004). *Windowsi, Internet, softver, multimedija, hardver* [Specijalno izd.]. Zagreb : Bug.
2. Brinch Hansen, P. (2001). *Classic Operating Systems: From Batch Processing to Distributed Systems*. New York: Springer.
3. Budin, L., Golub, M., Jakobović, D., Jelenković, L. (2010). *Operacijski sustavi.* Zagreb: Element.
4. Car, D. (2015). *Građa i upotreba računala.* Zagreb : Algebra.
5. Hrgarek, N. (2006). *Upravljanje resursima i problemima za unapređenje kvalitete programske opreme*
6. Jelenković, L. (2018). *Operacijski sustavi.* Sveučilište u Zagrebu, Fakultet elektrotehnike i računarstva. http://www.zemris.fer.hr/~leonardo/os/fer/OS-skripta.pdf. [pristupljeno 25. kolovoza 2019].
7. Kovačić, B. (2008) *Operacijski sustavi.* https://www.studytonight.com/operating-system/types-of-os. [pristupljeno 19. kolovoza 2019].
8. P. Avgeriou, D. Shepherd (2018). *Journal of Systems and Software*
9. PadaKuu. *Operating System Types.* (2017). http://www.padakuu.com/article/26-types-of-operating-systems-batch-operating-system-time-sharing-systems-distributed-os-network-os-real-time-os [pristupljeno 01. rujna 2019].
10. Povijest računala, https://racunalapc.weebly.com/povijest-ra269unala.html. [pristupljeno 27. kolovoza 2019].
11. Ribarić, S. (2011). *Građa računala: arhitektura i organizacija računarskih sustava.* Zagreb: Algebra.
12. Silberschatz, A., Galvin, B. and Gagne, G. (2012). *Operating System Concepts.* Addison Wesley.
13. Sterling, T., Anderson, M. and Brodowicz, M (2018). High Performance Computing
14. Tanenbaum A., Bos, H. (2015). *Modern Operating Systems.* Amsterdam.
15. Tanenbaum, A. (1997). *Operating Systems, Design and Implementation.* Prentice Hall.
16. Tanenbaum, A. S. (1993). Distributed operating systems anno 1992. What have we learned so far?. *Distributed Systems Engineering.* 1(1), 3–10
17. thePC.co, https://thepc.co/ [pristupljeno 30. kolovoza 2019].

Popis slika