Algoritmi raspoređivanja procesa operacijskog sustava

Gajari, Filip

SVEUČILIŠTE JOSIPA JURJA STROSSMAYERA U OSIJEKU
FAKULTET ELEKTROTEHNIKE, RAČUNARSTVA i INFORMACIJSKIH TEHNOLOGIJA U OSIJEKU

Sveučilišni preddiplomski studij računarstva

ALGORITMI RASPOREĐIVANJA PROCESA OPERACIJSKOG SUSTAVA

Završni rad

Filip Gajari

Osijek, 2018.

SADRŽAJ

1. UVOD

Proces predstavlja računalni program koji se izvršava. Sadrži programski kod procesa te njegovu trenutnu aktivnost. Proces može biti sastavljen od više niti (eng. *threads*) izvođenja dok broj niti od kojih se on sastoji ovisi o operacijskom sustavu. Niti su dijelovi procesa koji predstavljaju izvršavanje nekog dijela programa unutar adresnog prostora procesa te izvršavaju upute istodobno.

Pojavom multiprogramiranja postignuto je istovremeno korištenje procesora od strane više procesa, ali su se pojavili i neki problemi kao što su zastoji. Kako bi se ti zastoji spriječili, koriste se postupci raspoređivanja.

Raspoređivanje je postupak izrade rasporeda kojim će se procesi izvoditi na procesoru. Kod multiprogramiranja postoji više procesa koji istovremeno zahtijevaju procesor pa je potrebno promijeniti algoritme raspoređivanja kojima se raspoređuje izvršavanje tih procesa. Algoritmima raspoređivanja sprječava se nastanak zastoja odnosno situacija iz kojih procesi ne mogu izaći nakon međudjelovanja dvaju ili više procesa.

1.1. Zadatak završnog rada

Zadatak završnog rada je napraviti algoritme raspoređivanja i testirati iste, te prikazati izvršavanje procesa na vremenskoj listi. Za izradu algoritama bio je odabran programski jezik C. Algoritmi koji su izrađeni spadaju u skupine raspoređivanja u batch sustavima (FCFS, SJF i SRTN) i raspoređivanja u interaktivnim sustavima (RR, PS, SPN i FS).

2. PROGRAMI U IZVOĐENJU (PROCESI)

Procesi se mogu definirati kao programi koji se izvršavaju unutar računalnog sustava. Razlika u računalnim programima i procesima je ta što se računalni programi odnose na instrukcije koje se izdaju od strane korisnika računala dok su procesi vezani za izvršavanje instrukcija. Program može imati više procesa. To znači da ako se jedan program pokrene više puta, više procesa vezanih za taj program će se izvršiti [13].

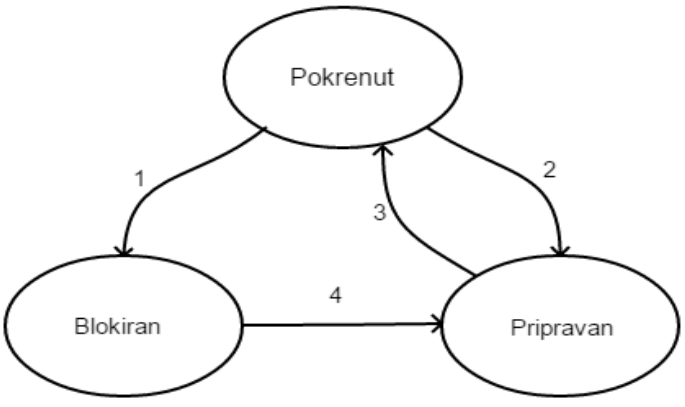
Procesi omogućuju korisniku pokretanje više programa istovremeno kako bi procesor ostao iskorišten čekajući na ulaze. Dijeljenje vremena (eng. *time-sharing*) omogućuje različitim procesima pokretanje zadatka i stavljanje zadatka na čekanje. Vrijeme, od zaustavljanja jednog zadatka do izvođenja drugog, je vrlo kratko.

Neki operacijski sustavi dijele procese na niti. Ono što niti obavljaju ovisi o procesu koji ih pokreće. Kada se procesi izvrše, niti se također izvrše. Procesi koji koriste veliki kapacitet procesora usporavaju funkcioniranje sustava u cjelini.

2.1. Stanja procesa

Kao što je prikazano na slici 1., procesi mogu imati 3 stanja:

1. pokrenut: trenutno koristi procesor,
2. pripravan: pokretljiv, ali zaustavljen kako bi se moga izvesti drugi proces,
3. blokiran: ne može se pokrenuti dok ne dođe do nekog vanjskog događaja.



Slika 1. Stanja procesa

Na istoj slici objašnjeni su i međukoraci:

1. nema ulaznog podatka pa se proces ne može pokrenuti,
2. raspoređivač oduzima procesor trenutnom procesu i dodjeljuje ga nekom drugom procesu,
3. raspoređivač vraća procesor procesu koji je bio aktivan prije trenutnog procesa,
4. proces se pokrenuo jer se dogodio neki vanjski događa.

2.2. Kriteriji za dodjelu procesora

Prije odabira algoritma koji će se koristiti za odabir procesa koji će dobiti procesor na korištenje, vrlo je važno odrediti kriterije kojima se uspoređuju različiti algoritmi. Kao što je navedeno u [1], cilj je postići što veću iskoristivost sustava uz što manja vremena proračuna, čekanja i odziva.

Kriteriji usporedbe algoritama raspoređivanja su:

1. Iskoristivost procesora (eng. *CPU utilization*): potrebno je držati procesor što više zaposlenim; općenito, iskoristivost procesora može biti između 0% i 100% međutim u stvarnim sustavima prosječna iskoristivost varira između 40% kod slabije opterećenih sustava do 90% kod jako opterećenih sustava
2. Propusnost (eng. *Throughput*): broj poslova odnosno procesa koji se obrade u jedinici vremena
3. Vrijeme obrade (eng. *Turnaround time*): vrijeme koje je potrebno procesoru da obradi neki proces, a predstavlja interval od trenutka kad proces dođe na obradu do trenutka kad se ta obrada završi
4. Vrijeme čekanja (eng. *Waiting time*): predstavlja ukupno vrijeme koje proces provede u redu pripravnih procesa
5. Vrijeme odziva (eng. *Response time*): vrijeme proteklo od trenutka kada je korisnik započeo proces do trenutka kada je proces dao prve rezultate

2.3. Algoritmi raspoređivanja

Algoritmi raspoređivanja se mogu podijeliti u 3 skupine: raspoređivanje u *batch* sustavima, raspoređivanje u interaktivnim sustavim i raspoređivanje u sustavima stvarnog vremena.

Ovisno kojoj skupini algoritama pripadaju karakteristični su po nekim ciljevima:

1. *Batch* sustavi:
    - propusnost – maksimalizirati broj poslove po satu,
    - vrijeme proračuna – minimalizirati vrijeme između podnošenja (eng. *submission*) i završetka (eng. *termination*),
    - iskoristivost procesora – održavanja procesora stalno zaposlenim.
2. Interaktivni sustavi:
    - vrijeme odziva – brz odziv na zahtjev,
    - razmjernost – zadovoljiti korisnikova očekivanja
3. Sustavi stvarnog vremena:
    - završetak poslova – izbjeći gubitak podataka,
    - predvidljivost – izbjeći gubitak kvalitete u multimedijskim sustavima
4. Svi sustavi (zajedničko za sve sustave):
    - jednakost – svakom procesu dodijeliti jednaku količinu procesora
    - provođenje politike (eng. *policy enforcement*) ,
    - ravnoteža – održati sve dijelove sustava zaposlenima.

Algoritmi koji spadaju u pojedine skupine su:

1. algoritmi raspoređivanje u *batch* sustavima:
    - „Prvi koji dođe prvi se posluži“ (eng. *First-Come First-Served*, FCFS),
    - „Najkraći posao prvi“ (eng. *Shortest Job First*, SJF),
    - „Najkraće preostalo vrijeme sljedeće“ (eng. *Shortest Remaining Time next*, SRTN).
2. algoritmi raspoređivanje u interaktivnim sustavim:
    - „Kružno dodjeljivanje“ (eng. *Round robin*, RR),
    - „Raspoređivanje po prioritetima“ (eng. *Priority Scheduling*, PS),
    - „Najkraći proces sljedeći“ (eng. *Shortest Process Next*, SPN,
    - „Ravnomjerno raspoređivanje“ (eng. *Fair-share Scheduling*, FS)
    - „Garantirano raspoređivanje“ (eng. *Guaranteed Scheduling*, GS)
    - Višestruki redovi.
3. algoritmi raspoređivanje u sustavima stvarnog vremena koji koriste napredne algoritme raspoređivanja

3. RASPOREĐIVANJE U BATCH SUSTAVIMA

Kod serijskih ili skupnih (eng. *Batch*) sustava, korisnici ne komuniciraju izravno s računalom. Svaki korisnik priprema svoj posao odnosno program na uređaju izvan mreže (eng. *offline),* kao što je univerzalna serijska sabirnica (eng. *Universal Serial Bus,* USB), i podnaša ga računalnom operatoru (koji je dio računala). Operator sortira programe sličnih karakteristika u iste skupine. Kako bi se proces ubrzao, programi koji imaju slične karaktristike, zajedno se raspoređuju (grupiraju) te se oni pokreću kao grupa.

Prednosti *batch* sustava su:

- proces se može izvesti i prilikom odsutnosti osobe, ako nekom poslu treba jako puno vremena (1 dan ili više),
- ne zahtijeva posebno računalno sklopovlje (eng. *Hardware*) i sustavnu podršku za unos podataka.

Nedostaci *batch* sustava su:

- teško je ukloniti pogreške (eng. *Debug*),
- nedostatak interakcije između korisnika i operacijskog sustava,
- u slučaju pojave pogreške, svi preostali poslovi su pogođeni te trebaju čekati da se pogreška ukloni,
- nema prekida u izvršavanju procesa ili ih ima malo.

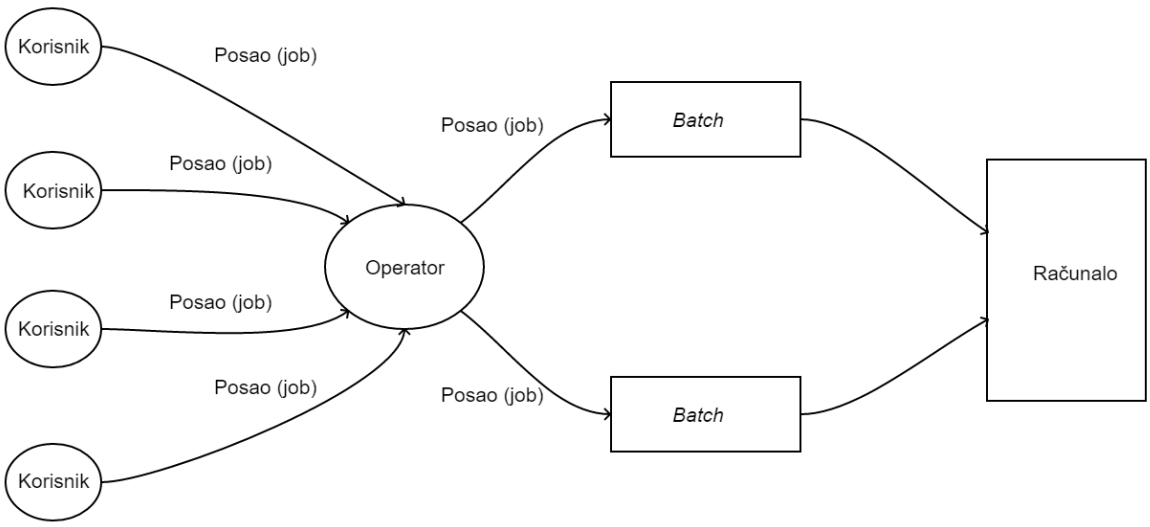
Kao što je navedeno u [2], *batch* poslovi su programi koji su dodijeljeni računalu te se dalje obavljaju bez prisustva korisnika. Primjeri *batch* poslova su zahtjev ispisa (na računalu) i analiza prijavljivanje (eng. *log in)* na internetsku stranicu.

Također treba spomenuti i *batch* obradu poslova. Ona se odnosi na rad računala kroz red (eng. *Queue)* ili seriju odvojenih programa bez ikakvog ručnog posredovanja. Ovisno o situaciji, svaki posao može imati povezane meta podatke kao što su klijent, odjeljenje ili korisnik, te pokazatelj prioriteta i zahtijevanih resursa.

Algoritmi raspoređivanja koji se ubrajaju u raspoređivanje u *batch* sustavima su:

- *First-Come First-Served*,
- *Shortest Job First*,
- *Shortest Remaining Tme Next*.

Slika 2 prikazuje osnovni tok izvođenja u *batch* sustavima:



Slika 2. *Batch* sustav

3.1. *First-Come First-Served* (FCFS)

First-Come First-Served predstavlja najjednostavniji ne prekidajući (eng. *non-preemptive)* algoritam raspoređivanja. Zasniva se na jednostavnom principu reda, *First In First Out* (FIFO). Kao što je prikazano na slici 3, prvi proces stavlja se na početak reda te se izvršava bez prekida. Svaki sljedeći proces stavlja se na kraj reda i čeka svoj red izvršavanja. Također, ako neki proces ostane blokiran stavlja se na kraj reda. Proces koji prvi zahtjeva procesor izvršava se prvi, pri čemu onda ide drugi proces, treći proces itd. Svaki se proces izvršava puno vrijeme trajanja bez ikakvog prekida.

Prosječno vrijeme obrade kod FCFS je dugačko u odnosu na ostale algoritme raspoređivanja iste skupine.

Prednosti FCFS su:

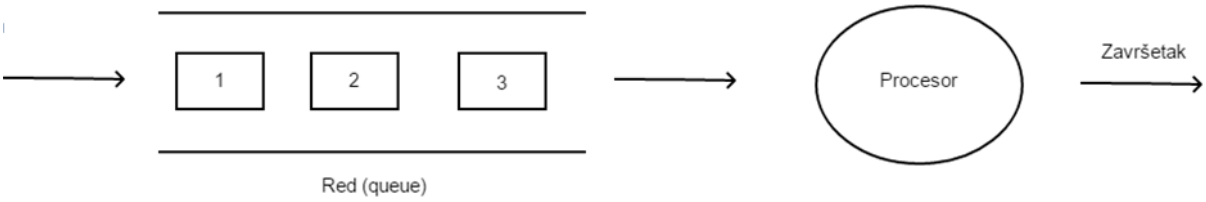
- jednostavnost izvršavanja,
- jednostavno praćenje procesa i njihovog izvršavanja,
- jednostavan za implementaciju.

Nedostaci FCFS su:

- tijekom izvođenja, ulazno/izlazni uređaj je besposlen,

- prosječno vrijeme izvršavanja je dugačko,
- procesi sa manjim vremenom izvršavanja „pate“ (predugo čekaju na izvršenje),
- favorizira procese vezane za procesor (eng. *CPU bound*[1]), a ne vezane za ulaz/izlaz (eng. *I/O bound*[2]).

Slika 4. prikazuje izvršavanje procesa prema FCFS-u:



Slika 3. Izvršavanje procesa korištenjem FCFS algoritma

3.2. *Shortest Job First* (SJF)

Shortest Job First (SJF), predstavlja jednostavan, ne prekidajući, *non-preemptive* algoritam raspoređivanja koji odabire onaj proces sa najkraćim djelovanjem. U slučaju da dva procesa traju jednako, primjenjuje se FCFS algoritam te se odabire onaj proces koji je prije došao.

SJF algoritam je optimalan i daje minimalno prosječno vrijeme čekanja za neki skup procesa što je navedeno pod [3].

Također postoji i *preemptive* verzija *Shortest Job First* algoritma koja se zove S*hortest Remaining Time Next.*

Prednosti SJF algoritma su:

- optimalan je sve dok su svi procesi dostupni na početku,
- uvijek izvršava proces čije je vrijeme trajanja najkraće,
- prosječno vrijeme čekanja kod SJF je manje nego kod FCFS.

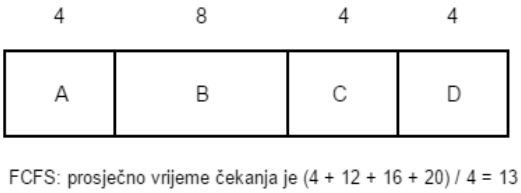
Nedostaci SJF algoritma su:

- kako raspoređivač izračuna koliko dugo će trebati procesu da se izvrši,

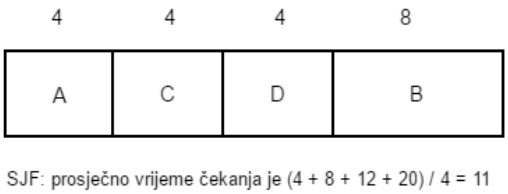
[1] Vrijeme potrebno za dovršavanje nekog zadatka određeno brzinom središnje procesorske jedinice (*eng. CPU*)
[2] Vrijeme potrebno za dovršavanje nekih izračuna određeno vremenom potrošenom na čekanje dovršenja ulazno/izlazbe operacijeStanje u kojem je vrijeme potrebno za dovršavanje izračuna određeno vremenom potrošenom na čekanje da se ulazno/izlazne operacije dovrše

- kada ima puno kratkih poslova, dulji poslovi su blokirani neodređeno vrijeme (*eng. starvation* [3]).

Bitna razlika između FCFS i SJF je u tome što je prosječno vrijeme čekanja kod SJF kraće nego kod FCFS. To se može vidjeti slici 4a i 4b.



Slika 4a. Prosječno vrijeme trajanja kod FCFS-a



Slika 4b. Prosječno vrijeme trajanja kod SJF-a

3.3. *Shortest Remaining Tme Next* (SRTN)

Shortest Remaining Time Next (SRTN), predstavlja prekidajuću verziju *Shortest Job First* algoritma. Ovaj algoritam izvršava procese sa najkraćim vremenom trajanja sve dok ne dođe sljedeći proces; neovisno o njegovom trajanju. Trenutni proces staje sa izvršavanjem, te se izračunava ukupno vrijeme trajanja (eng. *Total time*). Na temelju ukupnog vremena trajanja odabire se najkraći proces koji se tada isto izvršava do dolaska sljedećeg procesa. Kad ukupno vrijeme trajanja postane jednak vremenu dolaska procesa koji dolazi najkasnije, odnosno kada svi procesi postanu dostupni, tada se primjenjuje SJF algoritam.

STRN algoritam radi dobro sve dok procesi dolaze postupno. Kao i kod SJF algoritma vrijeme trajanja procesa mora biti poznato.

Prednosti SRTN algoritma su:

- predstavlja *preemptive* algoritam što smanjuje vrijeme čekanja u odnosu na *non-preemptive* verziju (SJF),
- kratki poslovi se izvršavaju jako brzo,
- vrijeme odziva je kratko.

Nedostaci SRTN algoritma su:

- veliko opterećenje procesora,
- vremenska dobit je smanjena zbog konteksta prebacivanja,
- kako raspoređivač izračuna koliko dugo će trebati procesu da se izvrši.

4. RASPOREĐIVANJE U INTERAKTIVNIM SUSTAVIMA

U interaktivnim sustavima korisnik i računalo su u interakciji. Izmjena informacija između korisnika i računala se događa stalno i dinamički. Kao što je opisano pod [4], priroda interaktivnih sustava najbolje se može razumjeti kroz Normanov model procjene/izvršenja (eng. *Norman's evaluation/execution model*) koji je prilagođen navedenoj temi:

- korisnik ima cilj,
- korisnik pokušava shvatiti kako izvršiti skup zadataka koji će ga odvesti do željenog cilja,
- korisnik obavlja neke akcije kao što su unošenje ulaznih podataka u sustav pritiskanjem gumba, dodirivanjem ekrana ili riječima,
- korisnik promatra rezultate svoje aktivnosti i pokušava procijeniti jesu li njegovi ciljevi ostvareni ili ne.

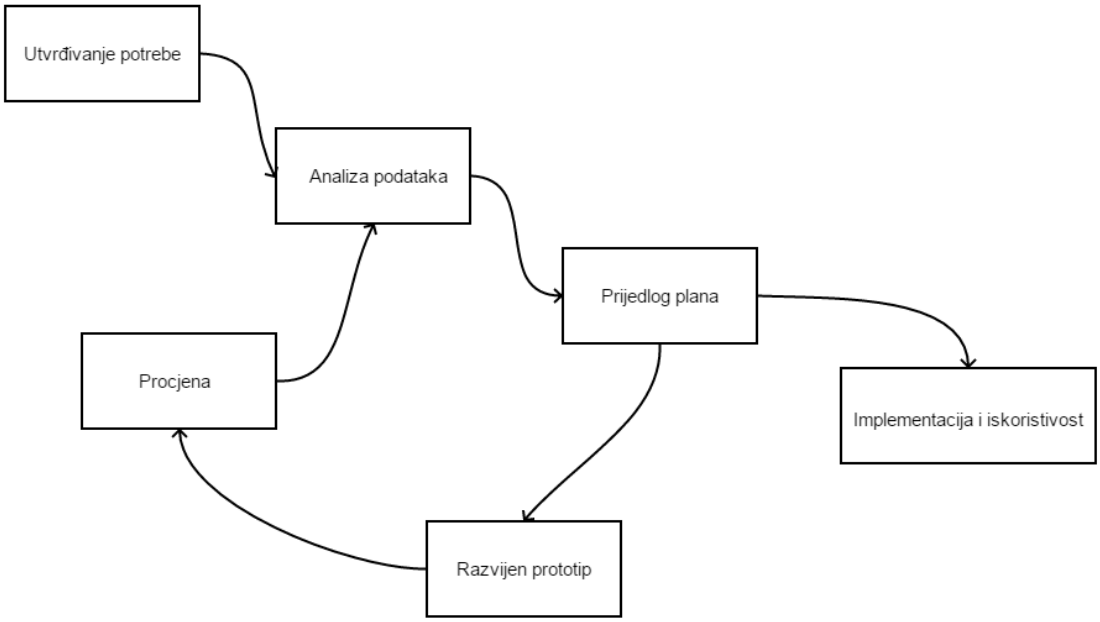
Priroda interaktivnih sustava odnosno životni ciklus interaktivnih sustava može se vidjeti na slici 5. Na njoj su prikazane faze Normanovog modela procjene/izvršenja.

Dobri interaktivni sustavi omogučuju korisnik vrlo lagano upravljanje sustavom, s ciljem postizanja određenih ciljeva, ali i mogučnost lagane procijene rezultate svoje aktivnosti.

Kada se govori o pojmu interaktivnim sustavima, on se može primijeniti i na neke druge uređaje (osim računala): mobilne telefone, navigacijske sustave u automobilima, video snimače, bankomate, *World Wide Web* itd.

Algoritmi raspoređivanja koji se ubrajaju u raspoređivanje u *batch* sustavima su:

- *Round Robin,*
- *Priority Scheduling,*
- *Shortest Proces Next,*
- *Fair-share Scheduling,*
- *Guaranteed Scheduling,*
- Višestruki redovi.



Slika 5. Životni ciklus interaktivnih sustava

Algoritmi raspoređivanja koji se ubrajaju u kategoriju raspoređivanja u interaktivnim sustavima su: *Round Robin*, *Priority Scheduling*, *Shortest Proces Next*, *Fair-Share Scheduling*, *Guaranteed Scheduling* i višestruki redovi.

4.1. *Round Robin* (RR)

Round Robin predstavlja jedan od najstarijih, najjednostavnijih, najpravednijih i najčešće korištenih algoritama raspoređivanja. U osnovi je zasnovan na FCFS algoritmu, ali je dodano vremensko ograničenje odnosno interval u korištenju procesora.

Vremenski interval se naziva kvant (eng. *quantum*) i dodjeljuje se svakom procesu, unutar kojeg se ima pravo izvršavati. Vremenski intervali su obično između 10 i 100 ms [5].

Round Robin je jednostavan za implementirati. Vrlo je važno da raspoređivač održi listu procesa koji su pokretljivi kao što je pokazano na slici 6. Ako se proces ne završi u tom intervalu, prekida se i procesor se dodjeljuje nekom drugom procesu. Kada proces iskoristi svoj kvant stavlja se na kraj reda što se može vidjeti na slici 7.

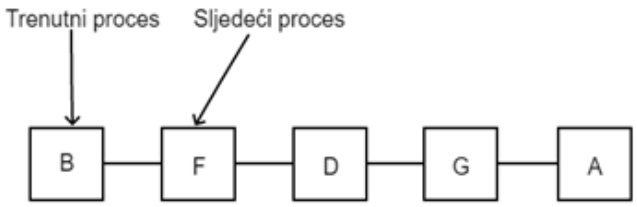
Problem *Round Robin*-a je duljina kvanta. Premještanjem s jednog procesa na drugi zahtjeva određenu količinu vremena za obavljanje administracije-uštede, punjenje registara,

memorijsko mapiranje, ažuriranje različitih tablica i lista, pražnjenja i ponovnog punjenja pred memorije (eng. *cache)*. Ako je vrijeme zamjene procesa 1 ms, a kvant je 4 ms, 20% procesora će biti „bačeno“ na *administration overhead*. Što je previše.

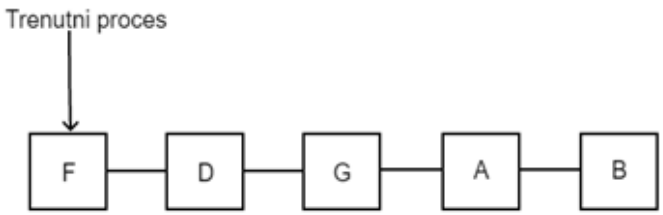
Kako bi se povećala efikasnost procesora, kvant se može postaviti na 100 ms pri čemu je izgubljeno vrijeme samo 1%. U slučaju da se 50 pojavi u vrlo kratkom vremenu svih 50 procesa bi bilo stavljeno u listu pokretljivih procesa. Ako je procesor besposlen, prvi proces će se odmah pokrenutim dok će drugi čekati 100 ms i svaki sljedeći još toliko. Zadnji bi tako trebao čekati 5 sekundi prije nego što se pokrene, ako se pretpostavi da su svi ostali iskoristili svoj cijeli kvant, a to je vrlo sporo za obavljanje takve kratke naredbe.

Drugi problem se javlja ako je kvant dulji od vremena kojeg svaki proces dobije na procesoru (eng. *CPU burst*). Pred pražnjenje se u ovom slučaju ne bi pojavljivalo jako često što bi dovelo do blokiranja procesa prije nego što bi kvant „iscurio“. To bi uzrokovalo zamjenu procesa.

Može se zaključiti da ako je kvant prekratak može doći do zamjene procesa i smanjenja efikasnosti procesora, ali ako je kvant prevelik može uzrokovati slab odgovor na kratke interaktivne zahtjeve. Optimalna veličina kvantuma je između 20 i 50 ms.



Slika 6: Redosljed izvršavanja procesa korištenjem RR algoritma (proces nije iskoristio kvant)



Slika 7: Redosljed izvršavanja procesa korištenjem RR algoritma (proces nije iskoristio kvant)

Prednosti RR algoritma raspoređivanja su:

- svakom procesu pridjeljuje se jednak dio procesora,
- jednostavnost.

Nedostaci RR algoritma raspoređivanja su:

- prosječno vrijeme čekanja može biti jako veliko u odnosu na ostale algoritme,
- opterećenje procesora veliko,
- ako je kvant prekratak može doći do smanjenja učinkovitosti procesora,
- ako je kvant predugačak može doći do slabog odziva na kratke procese.

4.2. *Priority Scheduling* (PS)

Priority Scheduling predstavlja raspoređivanje po razinama prvenstva. Svakom procesu su dodijeljene razine prvenstva odnosno prioritet koji se može mijenjati svakim novim otkucajem sata. Procesi unutar razina prvenstva najčešće se raspoređuju prema FCFS algoritmu. U slučaju da dva procesa imaju iste prioritete, prvo se odabire onaj koji je prvi došao.

Problem se javlja u slučaju visokih prioritete. Proces sa visokim prioritetima se mogu izvršavati neodređeno dugo[5]. Kako bi se to spriječilo, raspoređivač može smanjiti prioritet tog procesa svakim otkucajem sata. Time se postiže da prioritet tog procesa (procesa koji je imao visoki prioritet) padne ispod prioriteta sljedećeg procesa. To uzrokuje zamjenu procesa i sljedeći proces se može pokrenuti.

Alternativa tome je dodjeljivanje maksimalnog kvanta svakom procesu [6]. Kada se taj kvant iskoristi, pokrenuti proces staje i pokreče se sljedeći proces.

Prioriteti mogu biti statički ili dinamički. Statički prioriteti se dodjeljuju (eng. *allocated*) tijekom kreiranja te se ne mijenjaju, dok se dinamički prioriteti dodjeljuju ovisno o ponašanju procesa unutar samog sustava.

Prioriteti se također mogu definirati kao unutarnji (eng. *internally*) ili kao vanjski (eng. *externally*). Procesi s unutarnje definiranim prioritetima koriste mjerljivu količinu kako bi se odredili prioriteti danog procesa. Vanjski prioriteti se definiraju koristeći kriterije izvan operacijskog sustava kao što su značajnost procesa, tip procesa, korisnikove preferencije, izvore (resurse) koji se koriste za uporabu računala itd..

Prednosti PS algoritma raspoređivanja su:

- pruža dobar mehanizam definiranja važnosti svakog procesa,
- prioriteti.

Nedostaci PS algoritma raspoređivanja su:

- ako procesi visokih prioriteta koriste jako velik kapacitet procesora, procesi sa manjim prioritetima mogu biti odgođeni na neodređeno vrijeme,
- situacije kada se procesi nikada ne rasporede,
- određivanje prioriteta procesima, odnosno koji prioritet će se dodijeliti određenom procesu.

Razlikuju se dvije vrste *Priority Scheduling*-a:

1. *Preemtive Prioriy Scheduling*,
2. *Non-preemtive Priority Scheduling.*

4.2.1. *Non-preemptive Priority Scheduling* (NPPS)

Non-preemptive Priority Scheduling algoritam raspoređivanja raspoređuje procese na temelju prioriteta. U trenutku dolaska procesa u red čekanja, njegov prioritet se uspoređuje s prioritetima ostalih procesa koji su u redu čekanja, ali i s onim koje procesor u tom trenutku izvodi. Odabire se dostupni proces sa najvećim prioritetom te se izvodi do kraja, onoliko koliko mu je vrijeme trajanja. Kako bi proces uopće bio odabran, mora biti dostupan odnosno njegovo vrijeme pokretanja mora biti unutar *total time*-a koji se neprekidno izračunava kako se koji proces pokrene.

4.2.2. *Preemptive Priority Scheduling* (PPS)

Kod *Preemptive Priority Scheduling* algoritam raspoređivanja kao i kod *Non-preemptive Priority* algoritma, u trenutku dolaska procesa u red čekanja, njegov prioritet se uspoređuje s prioritetima ostalih procesa koji su u redu čekanja, ali i s onim koje procesor u tom trenutku izvodi. Također, ne mogu se odabrati procesi koji nisu dostupni odnosno čije vrijeme dolaska nije unutar *total time*-a koje se neprestano mijenja kako se koji proces pokrene. Proces s najvećim prioritetom koji se nalazi unutar ukupnog vremena odabire se među svim ostalim dostupnim procesima te sljedeći dobiva procesor. Vrijeme trajanja svakog procesa se smanjuje za vrijeme trajanja njegove aktivnosti do dolaska slijedećeg procesa.

U trenutku kada svi procesi postanu dostupni primjenjuje se *Non-preemtive* verzija algoritma. Procesi se raspoređuju prema prioritetima te se pokreću do kraja, onoliko koliko su raspoloživi.

4.3. *Shortest Proces Next* (SPN)

Shortest Proces Next predstavlja prilagođeni SJF algoritam raspoređivanja za interaktivne sustave. Kao i kod SJF algoritma, procesi se raspoređuju ovisno o njihovom trajanju. Ovaj algoritam odabire proces sa najkraćim djelovanjem koji se izvodi do kraja bez ikakvog prekida. Svaki put kad se neki proces pokrene izračunava se *total time*. Na temelju *total time*-a odabire se sljedeći najkraći dostupni proces.

4.4. *Fair-share Scheduling* (FS)

Fair-share Scheduling predstavlja algoritam raspoređivanja koji osigurava ravnomjernu raspodjelu procesora među svim korisnicima ili grupama, za razliku od ravnomjerne distribucije među procesima.

Ako imamo 4 korisnika koji u istom trenutku izvršavaju po jedan proces, raspoređivač će podijeliti procesor na način da svaki korisnik dobije 25% (100% / 4 = 25%) [6]. U slučaju da drugi korisnik pokrene svoj drugi proces, on će još uvijek imati dodijeljeno 25% procesora, ali će tada svaki pokrenuti proces tog korisnika (pokrenuta su dva procesa) koristiti po 12.5% od dozvoljenih 25%. Ako u trenutku kad su svi korisnici aktivni dođe novi korisnik tada će svaki korisnik dobiti 20% procesora (100% / 5 = 20%).

Također korisnici se mogu podijeliti u grupe. Procesor se u tom slučaju dijeli među grupama, a nakon toga među korisnicima unutar grupa. Uz pretpostavku da postoje 3 grupe i svaka grupa ima više korisnika raspodjela među grupama može se vidjeti na slici 7.

100% / 3 grupe = 33.3% po grupi

Grupa 1: (33.3% / 3 korisnika) = 11.1% po korisniku

Grupa 2: (33.3% / 2 korisnika) = 16.7% po korisniku

Grupa 3: (33.3% / 4 korisnika) = 8.3% po korisniku

Slika 8. Podjela procesora među grupama i korisnicima

5. RASPOREĐIVANJE U SUSTAVIMA STVARNOG VREMENA

Sustavi stvarnog vremena (eng. *real-time systems*) predstavljaju sustave koji obavljaju aplikacije u stvarnim sustavima. Oni obrađuju podatke u trenutku dolaska podatka u sustav. Sustavi ove kategorije sadrže ograničenja unutar kojih se podaci moraju obraditi. Ako se ograničenja ne poštuju, to dovodi do pada samog sustava.

Sustavi stvarnog vremena se dijele na „tvrdi“ i „mekane“ sustave. „Tvrdi“ sustavi (eng. *hard real-time systems*) stvarnog vremena imaju manje *jitter*-a dok „mekani“ sustavi (eng. *soft real-time systems*) stvarnog vremena imaju više [6].

Glavni cilj sustava stvarnog vremena je osiguranje *soft* i *hard* svojstava, a ne velika propusnost procesa koji se obrade u jedinici vremena.

Sustavi stvarnog vremena imaju napredne algoritame raspoređivanja. Ti algoritmi odnosno raspoređivači omogućuju široku računalno-sustavnu sinergiju prioriteta procesa, dok su sustavi stvarnog vremena češće usmjereni uskom skupu aplikacija. Glavni faktori su minimalna latencija za prekide (eng. *interrupt latency*[4]) i minimalna latencija za promjenu niti (eng. *thread switching latency*[5]), te imaju veću vrijednost kada brže i bolje mogu obaviti određeni broj poslova u nekom određenom vremenu [7]. Drugim riječima, sustavi stvarnog vremena su bazirani na kvaliteti, a ne na kvantiteti određenih brojeva poslova u određenom vremenu.

[4] https://en.wikipedia.org/wiki/Interrupt_latency
[5] https://en.wikipedia.org/wiki/Context_switch#LATENCY

6 . TESTIRANJE ALGORTAMA RASPOREĐIVANJA

Zadatak praktičnog dijela ovog rada bio je implementirati algoritme raspoređivanja i testirati iste. Okruženje koje je bilo korišteno za izradu algoritama bio je *Visual Sudio* 2015 i programski jezik C.

Svi algoritmi korisniku dopuštaju da unese broj procesa, vrijeme pojavljivanja i vrijeme trajanja procesa te kvanta ili prioriteta ako se radi o RR algoritmu odnosno PS algoritmu.

Svaki proces predstavljen je strukturom koja sadrži poziciju procesa, vrijeme pojavljivanja i trajanje procesa dok je kod nekih algoritama, koji to zahtijevaju, implementiran i red (eng. *queue*).

Procesi su korisniku nakon unosa navedenih parametra vidljivi u tablici koja prikazuje: broj procesa (PNO), vrijeme dolaska (AT) i vrijeme trajanja (BT) te prioritete (PR) u slučaju *Priority Scheduling*-a. U primjerima *Fair-Share Scheduling*-a procesi su u tablici dodijeljeni korisnicima (P) te je na kraju prikazana i dodjela procesora svakom korisniku (eng. *distribution of CPU*).

Rezultati izvršavanja procesa prikazani su na vremenskoj listi koja se može vidjeti na primjerima u potpoglavlju 6.2.

Svrha ovog rada bila je prikazati kako algoritmi funkcioniraju odnosno riješiti primjere te rješenja prikazati pomoću vremenske crte. Implementirani algoritmi mogu se koristiti u edukacijske svrhe odnosno kao pomoć pri učenju.

Nedostatak implementiranih algoritama raspoređivanja je taj što nisu optimizirani za veliki broj procesa.

6.1. Rezultati testiranja algoritama raspoređivanja

Testiranje je provedeno po algoritmima koji su implementirani u praktičnom dijelu rada. Odabrani su i prikazani oni testovi koji najbolje prikazuju osobine pojedinih algoritama raspoređivanja. Također, navedeni su i komentari koji su stečeni prilikom dodatnog testiranja.

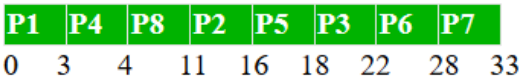
Implementacija algoritma korisniku nudi mogućnost unosa broja procesa, vremena dolaska procesa i vremena trajanja procesa preko tipkovnice. Kada korisnik unese parametre, program isporučuje željeni rezultat prikazan vremenskom listom.

6.1.1. First-Come First-Served

Tijekom testiranja programa primijećeno je da procesi sa manjim vremenom trajanja „pate“ te se sporo izvršava za broj procesa veći od 500.

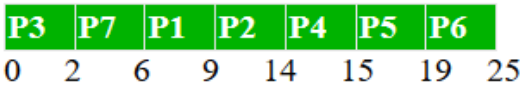
Na slikama ispod, obrađena su dva primjera. Na slici 9 prikazano je 7 procesa i obrađen je slučaj kada po 2 procesa imaju isti AT (prvo se pokreće proces koji je ima manji broj uz slovo P; dogovorno). Na slici 10 prikazano je 6 procesa sa različitim AT.

PNO	AT	BT
P1	1	3
P4	2	1
P8	2	7
P2	3	5
P5	3	2
P3	5	4
P6	5	6
P7	6	5



Slika 9. First-Come First Served 1

PNO	AT	BT
P3	0	2
P7	1	4
P1	2	3
P2	3	5
P4	4	1
P5	5	4
P6	6	6



Slika 10. First-Come First Served

6.1.2. *Shortest Job First*

Tijekom testiranja programa primijećeno je da je prosječno vrijeme čekanja malo, međutim u slučaju kada ima jako puno kraćih procesa, dulji procesi ostaju blokirani neodređeno dugo. Također, sporo se izvršava za broj procesa veći od 500.

Za ovaj algoritam obrađena su dva primjer. Na slici 11 prikazano je 7 procesa sa različitim AT (kako procesi postaju dostupni, pokreće se proces sa najmanjim AT od dostupnih procesa). Na slici 12 prikazano je 9 procesa od kojih neki procesi imaju isti AT (prikazano je kako algoritam rješava slučaj kada se procesi sa različitim vremenom trajanja pojavljuju u isto vrijeme).

PNO	AT	BT
P1	0	3
P2	2	5
P3	5	7
P4	6	8
P5	7	5
P6	9	6
P7	12	3

P1	P2	P5	P7	P6	P3	P4
0	3	8	13	16	22	29	37

Slika 11. Shortest Job First 1

PNO	AT	BT
P1	0	4
P2	1	5
P6	1	5
P7	3	1
P3	4	6
P4	4	6
P8	5	8
P5	6	2
P9	6	7

P1	P7	P2	P5	P6	P3	P4	P9	P8
0	4	5	10	12	17	23	29	36	44

Slika 12. Shortest Job First 2

6.1.3. *Shortest Remaining Time Next*

Tijekom testiranja programa primijećeno je da je se kraći procesi izvršavaju dosta brzo, te da se sporo izvršava za broj procesa iznad 500.

Za ovaj algoritam obrađena su dva primjer. Na slici 13 prikazano je 4 procesa sa različitim AT. Na slici 14 prikazano je 7 procesa sa različitim AT.

PNO	AT	BT
P1	0	8
P2	1	4
P3	2	9
P4	3	5

P1	P2	P2	P2	P4	P1	P3
0	1	2	3	5	10	17	26

Slika 13. Shortest Remaining Time Next 1

PNO	AT	BT
P1	0	7
P2	1	5
P3	2	3
P4	3	1
P5	4	2
P6	5	3
P7	7	1

P1	P2	P3	P4	P3	P3	P5	P5	P7	P6	P2	P1
0	1	2	3	4	5	6	7	8	9	12	16	22

Slika 14. Shortest Remaining Time Next 2

6.1.4. *Round Robin*

Tijekom testiranja programa primijećeno je da je se kraći procesi izvršavaju dosta brzo te se sporo izvršava za broj procesa veći od 200.

Za ovaj algoritam prikazana su dva primjera. Na slici 15 prikazan je slučaj kada je kvant 2 (optimalna veličina kvanta je između 20 ms i 50 ms), a na slici 16 prikazan je obrađen slučaj kada je kvant 3.

PNO	AT	BT
P1	0	4
P2	1	5
P3	2	2
P4	3	1
P5	4	6
P6	6	3

P1	P2	P3	P1	P4	P5	P2	P6	P5	P2	P6	P5
0	2	4	6	8	9	11	13	15	17	18	19	21

Slika 15. Round Robin 1

PNO	AT	BT
P4	1	9
P5	2	2
P3	3	7
P2	4	6
P1	5	5
P6	6	3
P7	7	4
P8	8	5

P4	P5	P3	P2	P4	P1	P6	P7	P8	P3	P2	P4	P1	P7	P8	P3
1	4	6	9	12	15	18	21	24	27	30	33	36	38	39	41	42

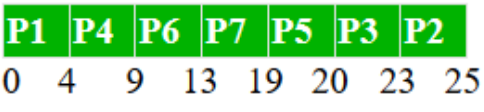
Slika 16. Round Robin 2

6.1.5. *Non-preemptive Priority Scheduling*

Tijekom testiranja programa primijećeno je da procesi velikih prioriteta mogu onemogućiti izvršavanje procesa manjih prioriteta, te se sporo izvršava za broj procesa veći od 300.

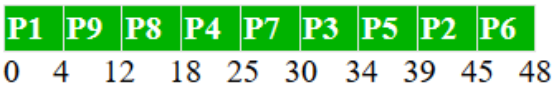
Za ovaj algoritam obrađena su dva primjera. Na slici 17 prikazano je 7 procesa, svi sa različitim PR. Na slici 18 prikazano je 8 procesa od kojih P2 i P4 imaju različiti PR (u ovom slučaju oba procesa su dostupna u isto vrijeme, ali se prvo pokreće onaj koji je prije došao).

PNO	PR	AT	BT
P1	2	0	4
P2	4	1	2
P3	6	2	3
P4	10	3	5
P5	8	4	1
P6	12	5	4
P7	9	6	6



Slika 17. Non-preemptive Priority Scheduling 1

PNO	PR	AT	BT
P1	1	0	4
P2	3	2	6
P3	5	3	4
P4	7	4	7
P5	4	5	5
P6	3	7	3
P7	6	4	5
P8	7	1	6
P9	8	2	8



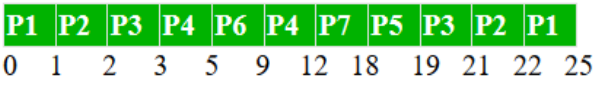
Slika 18. Non-preemptive Priority Scheduling 2

6.1.6. *Preemptive Priority Scheduling*

Tijekom testiranja programa primijećeno je da procesi velikih prioriteta mogu onemogućiti izvršavanje procesa manjih prioriteta te se sporo izvršava za broj procesa već od 300.

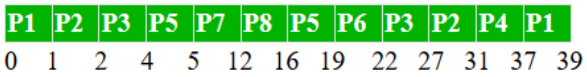
Za ovaj algoritam obrađena su dva ista primjera kao i na prijašnjem algoritmu (*Non-preemptive Priority Scheduling*). Na slici 19 prikazano je 7 procesa, svi sa različitim PR. Na slici 20 prikazano je 8 procesa od kojih P2 i P4 imaju različiti PR (u ovom slučaju oba procesa su dostupna u isto vrijeme, ali prvo se pokreće onaj proces koji ima manji AT). Kod ovog algoritma procesi se prekidaju ovisno o dolasku sljedećeg procesa.

PNO	PR	AT	BT
P1	2	0	4
P2	4	1	2
P3	6	2	3
P4	10	3	5
P5	8	4	1
P6	12	5	4
P7	9	6	6



Slika 19. Preemtive Priority Scheduling 1

PNO	PR	AT	BT
P1	1	0	3
P2	2	1	5
P3	3	2	7
P4	2	3	6
P5	5	4	4
P6	4	4	3
P7	7	5	7
P8	6	6	4



Slika 20. Preemtive Priority Scheduling 2

6.1.7. *Shortest Process Next*

Tijekom testiranja programa primijećeno je da algoritam funkcionira na istom principu kao i SJF međutim sporo se izvršava za broj procesa iznad 500.

Za ovaj algoritam obrađena su dva primjera. Na slici 21 prikazano je 7 procesa sa različitim AT (kako procesi postaju dostupni, pokreće se proces sa najmanjim AT od dostupnih procesa). Na slici 22 prikazano je 9 procesa od kojih neki procesi imaju isti AT (prikazano je kako algoritam rješava slučaj kada se procesi sa različitim vremenom trajanja pojavljuju u isto vrijeme).

Kao što je spomenuto, Shortest Process Next predstavlja prilagođeni Shortest Job First algoritam za interaktivne sustave.

PNO	AT	BT
P1	0	3
P2	2	5
P3	5	7
P4	6	8
P5	7	5
P6	9	6
P7	12	3

P1	P2	P5	P7	P6	P3	P4
0	3	8	13	16	22	29 37

Slika 21. Shortest proces Next 1

PNO	AT	BT
P1	0	4
P2	1	5
P6	1	5
P7	3	1
P3	4	6
P4	4	6
P8	5	8
P5	6	2
P9	6	7

P1	P7	P2	P5	P6	P3	P4	P9	P8
0	4	5	10	12	17	23	29	36 44

Slika 22. Shortest Proces Next 2

6.1.8. *Fair-share Scheduling*

Tijekom testiranja programa primijećeno je da algoritam svakom korisniku daje jednak dio procesora, ali se sporo izvršava za broj korisnika veći od 100.

Za ovaj algoritam obrađena su dva primjera. Na slici 23 može se vidjeti primjer sa 5 korisnika sa različitim brojem procesa. Na slici 24 može se vidjeti slučaj sa 4 korisnika sa istim brojem procesa.

P1:	A2	A3	A4		
P2:	B5	B2	B3	B4	
P3:	C5	C6	C1	C2	C3
P4:	D4	D6	D7		
P5:	E8	E4			

A2	B5	C5	D4	E8	A3	B2	C6	D6	E4	A4	B3	C1	D7	E8	A2	B4	C2	D4	E4	A3	B5	C3	D6	E8

```
Distribution of Cpu:
P1      : 6.666667
P2      : 5.000000
P3      : 4.000000
P4      : 6.666667
P5      : 10.000000
```

Slika 23. Fair-share Scheduling 1

P1:	A3	A2
P2:	B3	B2
P3:	C4	C5
P4:	D3	D5

A3	B3	C4	D3	A2	B2	C5	D5

```
Distribution of Cpu:
P1      : 12.500000
P2      : 12.500000
P3      : 12.500000
P4      : 12.500000
```

Slika 24 Fair-share Scheduling 2

7. ZAKLJUČAK

Tijekom rada na računalu može doći do različitih problema koji mogu usporiti rad procesora. Jedan od tih problema su zastoji. Kako bi se zastoji spriječili prije nego do njih dođe, koriste se algoritmi raspoređivanja koji na učinkovit način mogu napraviti plan raspoređivanja procesa. Ovisno o situaciji, ali i o sustavima, važno je odabrati odgovarajući algoritam raspoređivanja.

Kada se radi o interaktivnim sustavima, odgovarajući algoritmi raspoređivanja su *First-Come First-Served, Shortest Job First* i *Shortest Remaining Time Next* koji imaju kraće vrijeme odziva. Međutim, među navedenim algoritmima teško je odabrati optimalan. Svaki algoritam ima svoje prednosti i nedostatke. *First-Come First-Served* je jednostavan za implementaciju, ali izvršavanje kraćih procesa nije učinkovito. *Shortest Job First* algoritam je također jednostavan za implementirati te je izvršavanje kraćih poslova kod ovog algoritma učinkovito (za razliku od FCFS). Također u slučaju da dva procesa traju jednako, primjenjuje se *First-Come First-Served* algoritam što je dodatna prednost *SJF* algoritma. Međutim, iako je ovaj algoritam učinkovit za kraće procese, dulji procesi mogu biti blokirani na neodređeno vrijeme. *Shortest Remaing Time Next* je najsloženiji algoritam za implementaciju. Predstavlja prekidni oblik *SJF* algoritma, ali je i učinkovitiji. Nedostatak ovog algoritma je što uzrokuje veliko opterećenje procesora.

Kod sustava stvarnog vremena koriste se složeniji algoritmi koji se zasnivaju na prioritetima kao što je *Priority Scheduling*-a, podjeli korisnika u određene grupe kao kod *Fair-Share Scheduling*-a, ali i kvantu kao što je to slučaj kod *Round Robina*. *Round Robin* je jednostavan za implementaciju i svakom procesu daje jednak udio procesora. Ovaj algoritam je optimalan kad mu je zadan kvant u rasponu od 20 ms do 50 ms dok se za kvantume izvan tog raspona javljaju problemi. Iako je vrlo učinkovit algoritam, jako puno opterećuje procesor. *Priority Scheduling* algoritmi su učinkoviti jer se svakom procesu dodjeljuju prioriteti što znači da se procesi sa većim prioritetima prije izvršavaju od procesa s manjim prioritetima. Prekidna verzija PS algoritma je učinkovitija od neprekidne verzije jer dolazi do prekida procesa u slučaju dolaska procesa s većim prioritetom. Nedostatak *Priority Scheduling* algoritama je što procesi velikih prioriteta mogu okupirati procesor toliko dugo da se oni manjih prioriteta nikada ne izvrše. *Shortest Job Next* je identičan algoritam *Shortest Job First* algoritmu i radi na jednakom principu kao i on. *Fair-Share* algoritam je učinkovit kada postoji više korisnika sa više procesa te se korisnici dijele u grupe i svaki od njih dobiva jednak udio procesora.

8. LITERATURA

[1] Dodjela procesa, materijali s predavanja [online], Veleučilište u Rijeci, 2017., dostupno na https://www.veleri.hr/files/datotekep/nastavni_materijali/k_informatika_1/Predavanja_03.pdf, [svibanj, 2018.]

[2] SearchDataCenter, Batch [online], 2005, dostupno na https://searchdatacenter.techtarget.com/definition/batch, [svibanj, 2018.]

[3] CPU Scheduling, materijali s predavanja [online], Jordan university of Science and Technology, 2012., dostupno na https://www.uio.no/studier/emner/matnat/ifi/INF1060/h15/undervisningsmateriale/os-processes.pdf, [svibanj, 2018.]

[4] R. Griffiths, Norman's Gulfs of Execution and Evaluation, materijali s predavanja [online], University of Brighton, 2000., dostupno na http://www.it.bton.ac.uk/staff/rng/teaching/notes/NormanGulfs.html, [svibanj, 2018.]

[5] Scheduling in Interactive Systems, blog [online], Operating System Information Blog, 2013., dostupno na http://www.osinfoblog.com/post/100/scheduling-in-interactive-systems/, [svibanj, 2018.]

[6] Wikipedia [online], Wikipedia.org, dostupno na https://www.wikipedia.org/, [lipanj, 2018.]

[7] L. Apvrille, Operating Systems (III. Scheduling), materijali s predavanja [online], Universite Paris-Saclay, 2018., dostupno na http://soc.eurecom.fr/OS/docs/CourseOS_III_Scheduling.pdf, [svibanj, 2018.]

[8] A. Baveri, Operating Systems (CPU Scheduling), materijali s predavanja [online], Princeton University, 2010., dostupno na https://www.cs.princeton.edu/courses/archive/fall10/cos318/lectures/CPUScheduling.pdf, [svibanj, 2018.]

[9] Interactive Systems (Chapter 1. Interactive Systems), materijali za učenje, University of Cape Town, 2010., dostupno na https://www.cs.uct.ac.za/mit_notes/human_computer_interaction/htmls/ch01s03.html, [svibanj, 2018.]

[10] S. M. Vedantam, Batch operating systems, Quora [online], Gayatri Vidya Parishad College of Engineering for Women, 2017., dostupno na https://www.quora.com/What-is-a-batch-operating-system-time-sharing-operating-

system-distributed-operating-system-network-operating-system-and-an-embedded-system, [svibanj, 2018]

[11] P. Halvorsen, Opertaing Systems (Processes and CPU Scheduling), materijali s predavanja [online], University of Oslo, 2017., dostupno na https://www.uio.no/studier/emner/matnat/ifi/INF1060/h17/undervisningsmateriale/os-processes.pdf, [lipanj, 2018.]

[12] N. Sirpur, Shortest Process Next Program Explanation [online], YouTube.com, 2011., dostupno na: https://www.youtube.com/watch?v=TeC5AgWz4Hk , [svibanj, 2018.]

[13] Free resource libraries, What are computer processes?, lutilities.com [online], 2017., dostupno na http://www.liutilities.com/articles/what-are-computer-processes/#.W4kbdrgj5PY, [lipanj, 2018]

SAŽETAK

Proces predstavlja računalni program koji se izvršava. Kako ne bi došlo do zastoja između procesa, koriste se algoritmi raspoređivanja kojima se izrađuje plan raspoređivanja procesa na CPU-u.

Algoritmi raspoređivanja mogu se podijeliti u 3 kategorije: raspoređivanje u *batch* sustavim, raspoređivanje u interaktivnim sustavima te raspoređivanje u sustavima stvarnog vremena.

U ovom radu obrađeni su algoritmi koji spadaju u *batch* sustave i interaktivne sustave, te su implementirani i testirani isti. Također, kratko su spomenuti i sustavi stvarnog vremena.

ABSTRACT

Process represents a computer program that is being executed. In order to avoid interruptions between the processes, the scheduling algorithms used to create the CPU scheduling plan are used.

The scheduling algorithms can be divided into 3 categories: scheduling in batch systems, scheduling in interactive systems, and scheduling in real-time systems

This paper deals with algorithms that fall into categories of scheduling in batch systems and scheduling in interactive systems and the algorithms of the same categories have been implemented. Also, the real-time systems are briefly mentioned.

ŽIVOTOPIS

Filip Gajari rođen je 14.04.1996. godine u Zagrebu. Osnovnu školu završio je u Križevcima, a opću gimnaziju „Fran Galović“ u Koprivnici. Školske godine 2015/2016. maturirao je, te upisao Fakultet elektrotehnike, računarstava i informacijskih tehnologija Osijek sveučilišta u Osijeku, smjer Računarstvo. Osim toga, završio je 4 razreda glazbene škole „Albert Štriga“ u Križevcima, instrument violina.