SVEUČILIŠTE U ZAGREBU

FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

ZAVRŠNI RAD br. 98

Istraživački sustav Plan 9

Marko Pletikosa

Voditelj: *dr.sc. Leonardo Jelenković*

Zagreb, svibanj 2008.

SVEUČILIŠTE U ZAGREBU

FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

ZAVRŠNI RAD br. 98

Istraživački sustav Plan 9

Marko Pletikosa

Voditelj: *dr.sc. Leonardo Jelenković*

Zagreb, svibanj 2008.

Sadržaj

1. Uvod

Plan 9 je istraživački operacijski sustav koji su dizajnirali tvorci UNIX-a, s namjenom da postane nasljednik UNIX-a. Razvoj je započeo kasnih 80ih godina prošlog stoljeća, a zadnje izdanje je objavljeno 2002. godine [1]. U računarskom svijetu je zapažen po zanimljivim idejama koje su u njemu ostvarene. Tako je na primjer, svima nam poznati, UNICODE [2] skup znakova originalno nastao na Planu 9. Cilj ovog rada je upoznavanje sa sustavom, konceptima i idejama koje se razlikuju od konvencionalnih sustava te mogućnostima primjene tih ideja u postojećim sustavima.

U drugom poglavlju je prikazan pregled sustava i implementacija ideja, u trećem poglavlju je okvirno objašnjen način programiranja i prevođenja koda za više platformi na sustavu. U četvrtom poglavlju je opisan praktični rad i dodatno je pojašnjeno ostvarivanje paralelizma i sinkronizacije. Na kraju je dan kratak opis projekata na koje je Plan 9 inspirirao i izravno utjecao.

2. Opis sustava Plan 9

Tijekom 1980ih pojavio se trend u računarstvu koji se udaljavao od centraliziranih i vremenski dijeljenih računala, prema mreži manjih, radnih stanica, tipično baziranih na UNIX-ima. Ovo je značilo i značajan gubitak u performansama, koji se, međutim, s vremenom smanjio[2].

Tijekom prelaska na osobne radni stanice, neke od njihovih mana su previđene. Operativni sustav koji se na njima izvodio, UNIX, je sam po sebi vremenski dijeljeni sustav koji je imao problema s usvajanjem ideja koje su nastale nakon njega, poput mrežne i grafičke podrške. Štoviše, na takvim sustavima je jako teško simulirati stare monolitne vremenski dijeljene sustave. Administriranje i troškovi upravljanja takvim sustavima su višestruko veći od onih na centraliziranim sustavima.

Plan 9 je mrežno distribuirani operacijski sustav nastao na leđima UNIX-a, razvijen od strane tvoraca UNIX-a, ali u mnogočemu različit od UNIX-a. Razvoj Plan-a 9 je započet tijekom kasnih 1980-ih kao pokušaj da se dobije najbolje od oba svijeta: Izgraditi sustav koji je centralno administriran, ali isplativ, jer je sagrađen od jeftinih računala kao njegovih sastavnih elemenata. Ideja je bila izgraditi monolitni vremenski raspodijeljeni sustav temeljen na radnim stanicama. Različita računala obavljala bi različite zadatke: mala i jeftina osobna računala bi služila kao terminali za spajanje na podatkovne i procesorske poslužitelje. Poslužitelji bi obavljali pohranu podataka i procesorski zahtjevne kalkulacije, te bi bili bazirani na više procesorskim sustavima. Ideja je bila "Stvoriti UNIX sustav od puno malih sustava, a ne sustav od puno malih UNIX-a".

Problemi s UNIX-om su bili preveliki za rješavanja, ali neke od njegovih dobrih ideja su prenesene, kao što je uporaba datotečnog sustava za pristup resursima, čak i uređajima koji tradicionalno nisu smatrani datotekama. Za Plan 9, ova je ideja realizirana uz pomoć novog mrežnog protokola, nazvanog 9P, koji omogućava računalima transparentni pristup datotekama na udaljenim sustavima. Iznad ove koncepcije je izgrađen sustav imenika koji omogućava korisnicima stvaranje personaliziranih pogleda na sustav i dostupne resurse i usluge. Ovo je ključna karakteristika koja razlikuje Plan 9 od tradicionalnog UNIX sustava. Korisnik izgradi

vlastiti pogled na sustav, recimo na svom kućnom računalu, i u mogućnosti je bez većih poteškoća taj isti sustav ponovo izgraditi na kojem god se terminalu s pristupom mreži nađe. Model se pokazao bogatijim nego što ga se početno smatralo: Uskoro su procesi, resursi datotečnog sustava, grafika, pa čak i mreža dijeljeni preko imenika, 9P protokola i mreže.

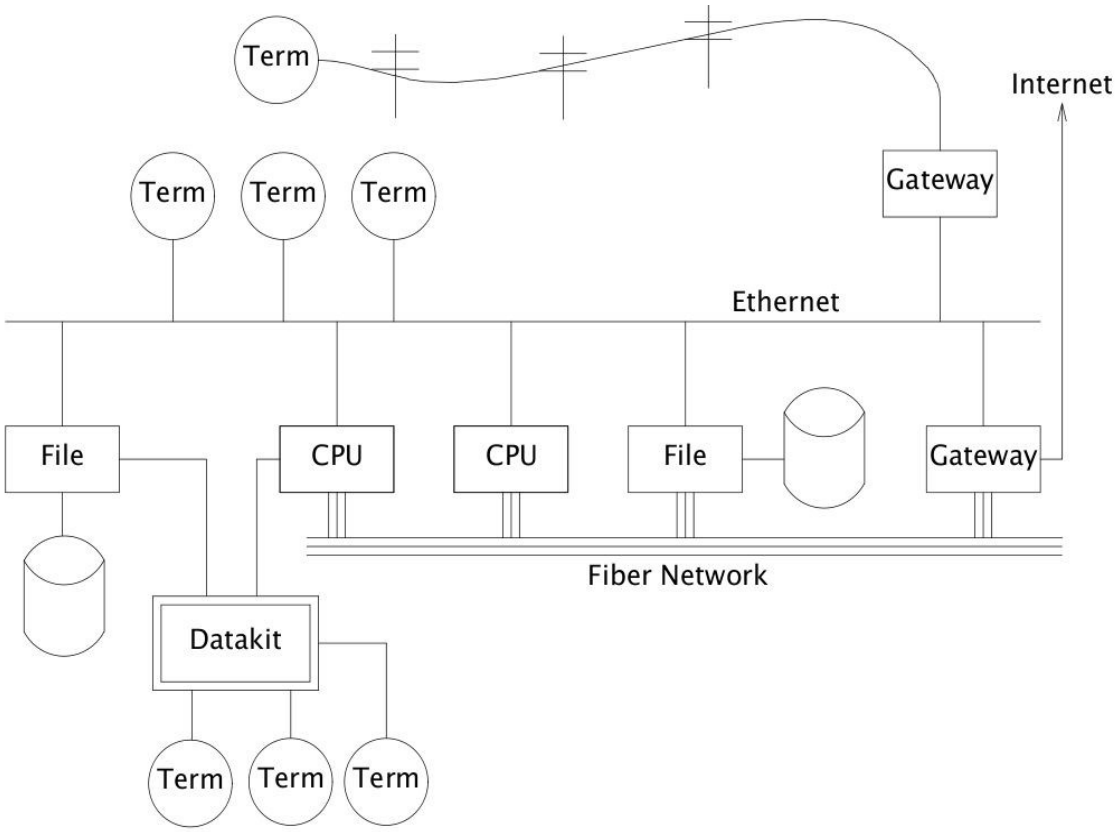
Mnogi problemi su riješeni, ne samo oni vezani uz jezgru operacijskog sustava UNIX-a, mnogi alati dostupni na UNIX-u su preneseni, dok su neki nanovo napisani. Plan 9 ima nove prevodioce, biblioteke, upravitelje grafikom i pregršt novih aplikacija. Da bi se omogućilo što lakše pisanje upravljačkih programa za sustav, izvorni kod je objavljen i dostupan svima.

2.1 Dizajn sustava

Sustav je temeljen na tri osnovna principa:

1. Resursi su imenovani i pristupa im se poput hijerarhijskom datotečnom sustavu,
2. 9P je standardan protokol za pristupanje tim resursima,
3. disjunktne usluge i hijerarhije se spajaju u zajedničke imeničke prostore.

Plan 9 kao sustav je pokušaj konzistentne i agresivne primjene ovih principa. Primjer instalacije sustava na kojoj se jako dobro mogu vidjeti sve prednosti dan je na Slici 1. Sustav se sastoji od više povezanih mreža, od kojih svaka pruža određenu funkcionalnost. Multiprocesorski sustavi obavljaju teže izračune, dok računala s većom količinom memorije pružaju pohranu podataka. Mreže između poslužitelja su visokopropusne optičke mreže. Kako se od poslužitelja krećemo prema terminalima, sve je manja potreba za brzom mrežom pa su terminali spojini konekcijama ostvarenima u ADSL ili ISDN tehnologiji. Svi terminali pružaju pristup resursima dostupnima na mreži, pa su u tom pogledu funkcijski ekvivalentni terminalima na starim vremenski podijeljenim sustavima. Ipak, kad korisnik koristi sustav, on stvara vlastiti imenički prostor s resursima koji su mu od interesa.



Slika 1. Prikaz strukture velike Plan 9 instalacije [2].

Umjesto da se koristi individualno sklopovlje za pojedine korisnike, koristi se personalizirani pogled na sustav od strane programske podrške. Personalizacija je omogućena stvaranjem imeničkog prostora unutar kojeg korisnik 'slaže' javno dostupne usluge koje želi koristiti. Pošto su najvažniji resursi na mreži datoteke, model sustava je datotečno orijentiran.

Sve usluge na mreži izvoze datotečne hijerarhije. Korisnik određuje one koje su mu od važnosti i njih stapa u svoj imenički prostor. Treba napomenuti da su usluge i njihovi nazivi uniformni, ali je pogled na sustav potpuno lokaliziran. Na primjer, ime */dev/cons* se uvijek odnosi na korisnikov terminal, no za svakog korisnika ovo ime predstavlja različite datoteke (npr. Za različite podržane procesorske arhitekture se koriste različite izvršne datoteke).

Glavnu poveznicu tako transparentnog sustava čini protokol 9P, kao strukturiran set transakcija koji šalje i prima poruke između klijenta i servera (lokalno i udaljeno). Protokol upravlja i datotečnim sustavom, ne samo datotekama. Korisnikov

imenički prostor je sadržan isključivo na lokalnom računalu, a pristup podacima je na razini okteta, ne blokova, za razliku od NFS-a i RFS-a.

Dizajn je temeljen s tradicionalnim datotekama na umu, ali se može koristiti i za mnoge druge resurse. Dobar primjer je datotečni sustav procesa, */proc,* koji omogućava jednostavan i čist pristup i kontrolu pokrenutim procesima. Model sustava je dobro prihvaćen od strane korisnika i dizajnera sustava, tako da je usluge temeljene na sučeljima kao datotekama jednostavno izgraditi, razumjeti i koristiti.

2.2 Pogled s komandne linije

. Plan 9 je namijenjen za korištenje na računalu s ekranom na kojim se izvodi upravitelj prozora *(eng. Window system)*. Upravljanje tipkovnicom je prilično rudimentalno, ali čim se na ekranu izvodi upravitelj prozora, u Plan 9 sustavu nazvan 8½ [8], onda se jednostavno uređuje tekst pomoću `cut` i `paste` operacija iz iskočivih menija, kopiranje između prozora i slično. 8½ omogućava uređivanje teksta iz prošlosti, ne samo iz trenutne linije unosa. Prema tvrdnjama autora, mogućnosti uređivanja dostupne pomoću 8½ bi trebale biti dovoljno snažne da kompenziraju nedostatke poput povijesti naredbi u školjki *(eng. Shell)* [6], nedostatka *scrollanja*, ali se po mom iskustvu u praksi nije pokazao kao dovoljno učinkovit.

Svaki prozor se stvara u zasebnom imeničkom prostoru, što za korisnika-početnika može biti malo zbunjujuće, pogotovo kad vidi više direktorija */dev.* Izmjene koje se naprave u jednom imeničkom prostoru nemaju utjecaja na druge prozore. Pozitivna strana ovakvog pristupa je mogućnost eksperimentiranja bez straha od trajnih negativnih posljedica. Čim se prozor izbriše, a njegove se promjene ne spreme, svi tragovi eksperimentiranja su izbrisani.

Svaki prozor izvodi jednu aplikaciju poput školjke sustava, s standardnim ulazom i izlazom spojenim na uredivi tekst prozora. Također, svaki prozor ima i multipleksiran pristup tipkovnici, mišu i drugim grafičkim resursima, poput */dev/mouse, /dev/bitblt* i */dev/cons* (analogno */dev/tty* na UNIX-u). 8½ je u suštini datotečni poslužitelj koji poslužuje prozore s navedenim i sličnim datotekama. Za razliku od X-windows-a koji za grafiku obično stvara novi prozor, 8½ izvodi grafičke aplikacije unutar istog prozora.

Originalna namjena Plan 9 sustava je izvršavanje interaktivnih aplikacija, poput uređivača teksta, izvršavanja proračuna, ili zahtjevnih operacija s podacima, na udaljenom poslužitelju. Različiti prozori se mogu izvršavati na različitim udaljenim računalima, ili lokalno, dok uporaba jedinstvenog imeničkog prostora za korisnika ovo čini potpuno transparentnim: iste naredbe i resursi su dostupni, s istim imenima, bez obzira gdje se obavlja izračunavanje.

Skup naredbi dostupan za Plan 9 je vrlo sličan onom dostupnom na UNIX-u. Neke od naredbi su u suštini identične onima na UNIX-u, poput ls, cat, awk, troff i sl. Neke od njih su iznova napisane, dok neke imaju jednostavniju implementaciju. S druge strane neke naredbe su potpuno nove, poput školjke *rc*, *debuggera acid,* ili tekst uređivača *sam*. Potonji je jako kompleksan i nejasan za početnike.

2.3 Datotečni poslužitelj

U mrežnoj instalaciji sustava, centralni datotečni poslužitelj poslužuje klijentska računala datotečnom hijerarhijom datoteka koristeći mrežni protokol 9P. Poslužitelj je orijentiran kao samostalan sustav kojem se može pristupiti isključivo preko mreže, dizajniran za obavljanje samo jednog posla. Ne izvodi nikakve korisničke procese, samo fiksni set rutina ugrađenih u *boot image.* Ta hijerarhija datoteka je dostupna kao jedinstveno stablo, bez obzira što se datoteke mogu nalaziti na više diskova.

Poslužitelj ima tri razine pohrane:

1. Spremnik *(eng. Buffer),*
2. magnetski disk *(eng. Magnetic disk),*
3. masovna pohrana (*eng. Bulk storage*), u koju se može zapisivati samo jednom

Prva razina pohrane je najmanja i ima najveću brzinu prijenosa. Svaka sljedeća razina ima veći kapacitet i manju brzinu. Sustav pravi periodičnu rezervnu pohranu na treću razinu pohrane, svako jutro u 05:00. Sustav se privremeno zamrzava dok se trenutna slika sustava ne zapiše na medij. Pohrana se sprema u */n/dump/yyyy/mmdd* direktorij i tamo je dostupna za čitanje, ali ne i za pisanje. Ovo je posebice korisno ako se žele pratiti promjene na izvornom kodu ili dokumentima s

vremenom. Druga uporaba ovakve pohrane je potpuna rezervna slika sustava. Bilo kakve teže pogreške načinjene sustavu se lako mogu poništiti. Negativna strana ovakvog pristupa je da se sve promjene na sustavu, nastale nakon datuma pohrane slike koja se dohvaća, su izgubljene. Prava pristupa datotekama se također spremaju pa tako nema straha da neovlašteni korisnici čitaju privatne dokumente. S druge strane, pošto se pohrane ne mogu uređivati, tako se ta prava ne mogu mijenjati.

2.4 Primjeri datotečnih poslužitelja

Plan 9 nema *ftp* naredbe. Umjesto nje, koristi se korisnički datotečni poslužitelj nazvan *ftpfs*, koji uspostavlja vezu s *ftp* poslužiteljem i autentificira se u ime korisnika i koristi FTP protokol za pregledavanje datoteka na udaljenom direktoriju. S druge stane, korisniku nudi hijerarhiju datoteka vezanu uz */n/ftp* direktorij. Drugim riječima, vrši se prevođenje FTP protokola u 9P protokol. Implementacija je nezgodna jer je potrebno obaviti prilično složeno upravljanje priručnom memorijom da bi se dobilo na učinkovitosti. No, rezultat se svakako isplati, jer uobičajene naredbe poput *cp, grep, diff* i slične rade s udaljenim datotekama kao da su one pohranjene lokalno, tj. s korisničke strane je transparentno. I drugi sustavi su imali slične ideje, no zbog jednostavnosti 9P protokola, u Planu 9 pristup je prirodniji.

Exportfs s druge strane, je korisnički proces koji dio vlastitog imeničnog prostora čini dostupnim drugim procesima prevodeći 9P zahtjeve u sistemske pozive Plan 9 jezgri. Datotečna hijerarhija kojoj omogućava pristup može sadržavati i datoteke s više drugih poslužitelja. *Exportfs* se obično koristi kao udaljeni poslužitelj kojeg pokreće lokalni program poput *import* ili *cpu. Import* poziva udaljeno računalo, tamo pokreće *exportfs* i spaja svoju 9P konekciju na lokalni imenični prostor. Primjer naredbe je:

import gunthor /proc

koja omogućava da se Gunthorovi procesi vide u */proc* datotečnom sustavu. Ovaj pristup omogućava lokalnim ispravljivačima programa *(eng. Debuggers)* pristup udaljenim procesima.

S druge strane, *cpu* naredba radi upravo suprotno: Spaja lokalni terminal s udaljenim CPU poslužiteljem – Nakon pozivanja poslužitelja, lokalno pokreće

exportfs i spaja ga s imeničnim prostorom procesa, obično novo stvorenom školjkom *(eng. Shell)* udaljenog sustava. Zatim preuređuje imenični prostor tako da omogući udaljenom računalu pristup upravljačkim datotekama lokalnih uređaja u poslužiteljskom */dev* direktoriju. Smisao ove naredbe je pokretanje školjke sustava na brzom računalu s imeničnim prostorom analognim lokalnom.

2.5 Konfigurabilnost, administracija i korisnici

Moguće je instalirati sustav na jedno računalo i imati samostojeći sustav, ali pošto je Plan 9 distribuirani operativni sustav, njegove prednosti se najbolje primjećuju u mrežnom okruženju. Namjena je imati centralne procesorske i podatkovne poslužitelje na koje se spajaju terminali.

Terminali ne pohranjuju (iako je moguće raditi i drugačije) podatke lokalno na disk, nego je sva pohrana namijenjena poslužitelju. Na ovaj način svi terminali su funkcionalno identični. Ovo svojstvo podsjeća na starije vremenski dijeljene sustave. Koristeći *BOOTP* i *TFTP*, jezgra sustava se prenosi preko mreže. Budući da postoji jedinstvena baza korisnika pohranjena na centralnom računalu, nema potrebe za sinkronizacijom različitih */etc/passwd* datoteka[2].

Ovako velikim stupnjem centralizacije zahtijeva se pouzdanost i brzina mreže za povezivanje računala, ali se omogućava znatno jednostavnija administracija, jer se centralni sustav nalazi na samo jednom računalu. Smanjuju se i troškovi unapređenja opreme, jer terminali ne obavljaju značajna izračunavanja – ulaže se u centralne poslužitelje opremljene s više procesora.

Plan 9 nema super-korisnika *(eng.super-user).* Svaki poslužitelj je odgovoran za održavanje vlastite sigurnosti, obično dozvoljavajući pristup samo preko školjke sustava, koja je zaštićena lozinkom. Poslužitelji imaju obično posebnog korisnika *adm* koji je zadužen za ažuriranje i administraciju sustava. Taj korisnik nema ovlasti za čitanje korisničkih datoteka, osim ako mu to korisnik ne dozvoli. Također, postoji i neovlašteni korisnik *none* koji nema lozinku. Svatko može tvrditi da je *none* i pristupiti sustavu, ali s puno manjim mogućnostima (ograničen imenički prostor) od regularnih korisnika. Ideja je slična anonimnim korisnicima FTP poslužitelja.

2.6 Implementacija imeničkih prostora

Kad god je to moguće, sustav se temelji na dvije jednostavne ideje: svaki resurs na računalu, bio on lokalan ili udaljen reprezentira se hijerarhijskom datotečnim sustavom; i korisnik ili proces stvara vlastiti privatni pogled na sustav pomoću imeničnog prostora koji spaja te resurse[4].

Svi resursi u Planu 9 izgledaju kao datotečni sustavi. Ipak, to ne znači da su to repozitoriji za trajno spremanje datoteka na disk, nego da je sučelje prema njima datotečno orijentirano. Pronalaženje datoteka (resursa), njihovo dodavanje po imenu, i pristupanje njihovom sadržaju se obavlja pomoću čitanja i pisanja datoteka. Postoje mnogobrojni datotečni sustavi u Planu 9, ali samo nekolicina njih prezentira tradicionalne datoteke. Na ovoj razini apstrakcije, datoteke su slične objektima, s razlikom da datoteke već imaju imena, pristup i metode zaštite pristupa.

Sučelje prema datotečnim sustavima je definirano pomoću protokola, 9P, analognog, ali ne previše sličnom NFS-u. Protokol se temelji na datotekama, ne blokovima – uz danu vezu prema korijenskom *(eng. Root)* direktoriju, 9P poruke navigiraju po datotečnom sustavu, otvaraju datoteke za čitanje/pisanje *(eng. I/O – input/output)*, i čitaju ili zapisuju podatke u datoteke. 9P sadrži 17 tipova poruka: tri za inicijalizaciju i autentifikaciju, i 14 za upravljanje datotekama. Te poruke su generirane od strane jezgre sustava, kao odgovor na korisnikove zahtjeve za čitanjem/pisanjem. *Auth* i *attach* autentificiraju vezu, uspostavljenu izvan protokola, i validiraju korisnika. Rezultat je autentificirani kanal *(eng. Channel)* koji upućuje na korijen datotečnog sustava. *Clone* napravi identičan kanal, koji može biti preusmjeren na druge datoteke koristeći *walk* za spuštanje jedan nivo u hijerarhiji. *Stat* i *wstat* čitaju atribute datoteke na koju upućuje kanal. *Open* otvara datoteku za predstojeće *read* i *write* koji čitaju i pišu u datoteku. *Create* i *remove* stvaraju i brišu datoteke, dok *clunk* prekida vezu kanala i datoteke, bez utjecaja na datoteku. Nijedna od 9P poruka ne služi za upravljanje priručnim spremnikom, za to su zadužene klijentska ili poslužiteljska strana.

Radi poboljšanja učinkovitosti, veza prema lokalnoj jezgri i njenim uređajima je ostvarena uobičajenim umjesto udaljenim pozivima funkcija. Funkcije se vežu jedan-na-jedan s 9P porukama. Lokalno svaki kanal je povezan sa strukturom podataka

koja sadrži polje tip-a koje se koristi za indeksiranje tablice poziva funkcija, jedan skup po tipu datotečnog sustava, analogno odabiru skupa metoda objekta. Jedan datotečni sustav unutar jezgre sustava, nazvan *mount device*, prevodi lokalne 9P pozive funkcije u RPC *(eng. Remote procedure call – poziv udaljene procedure ili funkcije)*, preko transportnog protokola poput TCP-a ili IL-a (više o IL protokolu će biti u nastavku rada), ili cjevovoda *(eng. Pipe)* do korisničkog procesa. *Write* i *read* pozivi prenose poruke preko preko transportnog sloja. *Mount device* je jedini most između proceduralnog sučelja koje vide korisnički programeri i udaljene i korisničke usluge. *Mount device* je u suštini posrednički *(eng. Proxy)* poslužitelj.

Korisnički procesi stvaraju imeničke prostore koristeći tri sistemska poziva: *mount, bind* i *unmount. Mount* spja datotečno stablo na trenutni imenički prostor. Prije pozivanja *mount* poziva, klijent mora stvoriti vezu s poslužiteljem i obliku opisnika datoteke (kanal) u koji se može čitati i pisati pomoću 9P poruka. Taj opisnik predstavlja cjevovod ili mrežnu poveznicu. Poziv *bind* kopira dio postojećeg imeničnog prostora u drugu točku prostora. Poziv *unmount* omogućava da se komponente prostora uklone.

Koristeći ili bind ili mount, moguće je više direktorija "naslagati" na jednu točku u prostoru – jedinstven direktorij. U terminologiji Plana 9, ovo se zove *union* direktorij i ponaša se kao da se unutar njega nalaze sve datoteke koje su u direktorijima od kojih je napravljen. Zastavica u pozivu bind i mount omogućuje specificiranje pozicije dodavanja novog direktorija uniji – na početak ili na kraj, ili čak potpuno zamijeniti uniju. Dok se datoteke direktorija-unije listaju, svaka komponenta se slijedno pretražuje. Unijski direktoriji su jedna od najkorištenijih organizacijskih značajki u Plan 9 imeničkom prostoru. Npr., direktorij */bin* je izgrađen kao unija */$cputype/bin, /rc/bin,* i eventualno korisnikovih direktorija . Time je uporaba varijable okruženja *$PATH* potpuno nepotrebna.

Postavlja se pitanje, u koji direktorij će se stvoriti/spremiti datoteka ako je stvorena u uniji direktorija. Tvorci Plana 9 su se odlučili da unijski direktoriji ne primaju nove datoteke, iako je dopušteno pozvati *create* nad postojećom datotekom.

2.7 Organizacija mreže

Mreža igra centralnu ulogu u svakom distribuiranom sustavu. Ovo je posebice točno za Plan 9, gdje je većina resursa ostvarena preko poslužitelja izvan jezgre sustava. Važnost mreža je vidljiva i u udjelu mrežnog izvornog koda u jezgri. Od početnih 25 000 linija koda, 12 500 je vezano uz mrežu i mrežne protokole. Implementacije protokola se sastoje većinom od sinkronizacije i upravljanja memorijom, područjima koja zahtijevaju istančane strategije upravljanja pogreškama.

Svaki upravljački program mrežnog sučelja je jezgreni datotečni sustav opisan s nekoliko datoteka[5]. Pogledajmo primjer s računala:

```
term% lc -l /dev/eia*
--rw-rw---- t 0 glenda glenda 0 Feb  5 15:41 eia0
--rw-rw---- t 0 glenda glenda 0 Feb  5 15:41 eia0ctl
--r--r--r-- t 0 glenda glenda 0 Feb  5 15:41 eia0status
--rw-rw---- t 0 glenda glenda 0 Feb  5 15:41 eia1
--rw-rw---- t 0 glenda glenda 0 Feb  5 15:41 eia1ctl
--r--r--r-- t 0 glenda glenda 0 Feb  5 15:41 eia1status
```

Kontrolna *ctl* datoteka upravlja uređajem: npr., upisivanjem vrijednosti b1200, postavlja se brzina prijenosa na 1200 .

Mrežna sučelja su također jezgreni datotečni sustavi. Podešavanje prekidanje veze je ostvareno upisivanjem tekstualnih poruka u kontrolnu datoteku povezanu s uređajem. Struktura i semantika uređaja je zajednička svim mrežama, tako da, osim zamjene imena datoteke, identični postupak uspostavlja vezu za različite mreže. Pogledajmo ispise za TCP protokol:

```
term% lc -l /net/tcp
d-r-xr-xr-x l 0 glenda  glenda 0 Feb  5 15:41 0          --rw-rw-rw- l 0 network glenda 0 Feb  5 15:41 clone
d-r-xr-xr-x l 0 glenda  glenda 0 Feb  5 15:41 1          --r--r--r-- l 0 network glenda 0 Feb  5 15:41 stats
d-r-xr-xr-x l 0 network glenda 0 Feb  5 15:41 2
```

Korijenski direktorij sadrži *clone* datoteku i direktorij za svaku od veza, numeriranih od 0 do n. Svaki direktorij predstavlja jednu TCP/IP vezu. Otvaranjem

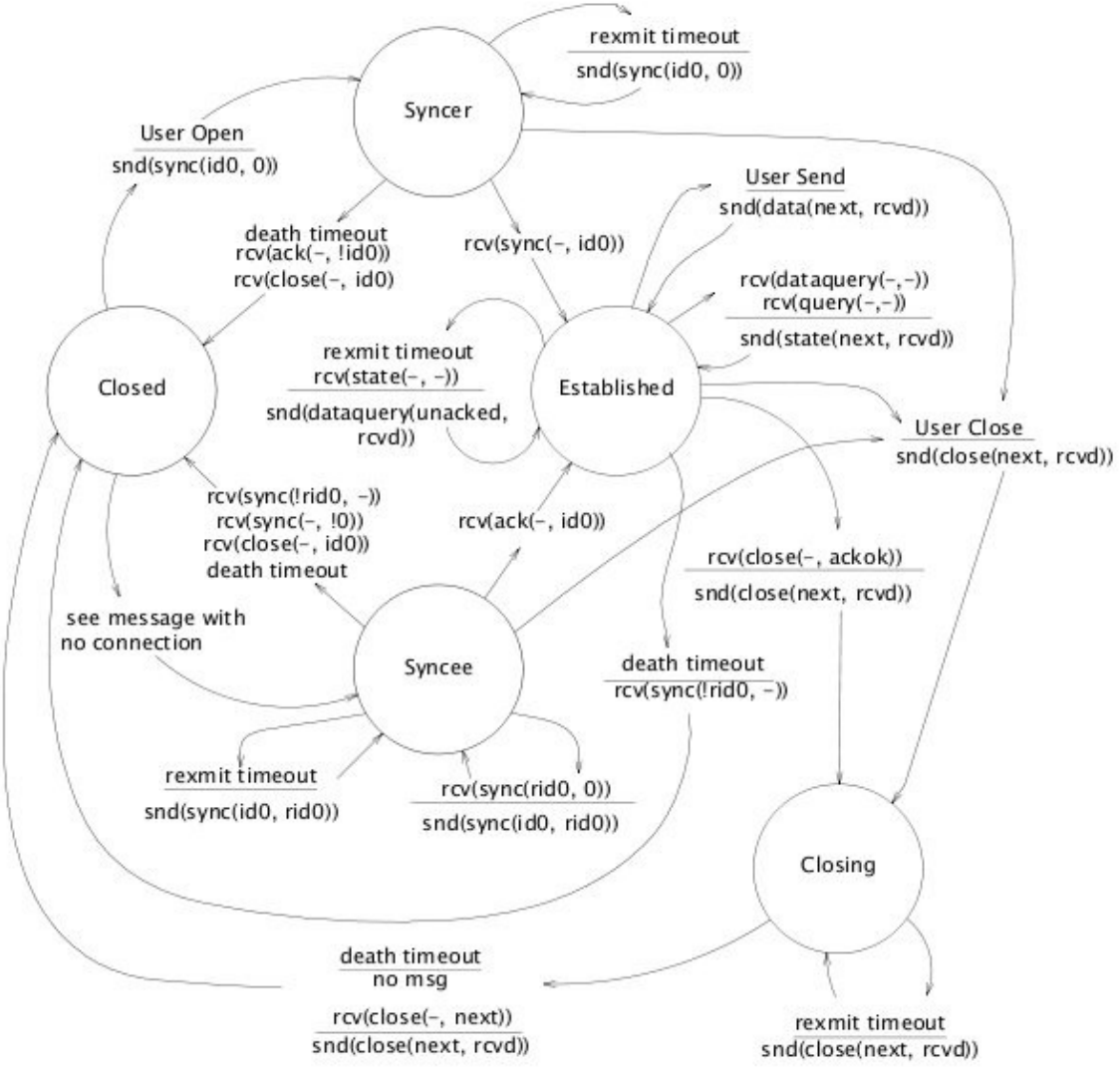
clone-a se rezervira nekorištena veza i vraća opisnik upravljačke datoteke. Čitanjem, pak upravljačke datoteke vraća tekstualni broj veze, tako da je s njom jednostavno stvoriti puno ime novog direktorija. *Local, remote* i *status* su dijagnostičke datoteke (npr. *remote* sadrži IP adresu i vrata udaljenog računala).

2.7.1 IL protokol

9P protokol se mora izvršavati iznad pouzdanog transportnog protokola s delimitiranim porukama. 9P nema mehanizme oporavka od pogreška u transmisiji i sustav pretpostavlja da svako čitanje iz komunikacijskog kanala vraća jednu 9P poruku; ne pretražuje tok podataka da bi otkrio granice poruke. Cjevovodi i neke mreže imaju ova svojstva, ali standardni IP protokoli nemaju. TCP ne delimitira poruke, dok UDP nije pouzdan.

Tvorci Plana 9 su dizajnirali novi protokol, nazvan IL *(eng. Internet Link)*, da bi omogućili slanje 9P poruka preko IP-a [13]. To je spojno orijentiran protokol koji omogućava pouzdanu transmisiju slijeda poruka. Pošto proces može imati samo jedan važeći 9P zahtjev, nema potrebe za kontrolom toka. U Planu 9, implementacija IL-a je jednostavnija i brža od TCP-a. Uspostava veze započinje dvosmjernim rukovanjem koje stvara inicijalne brojeve sekvence sa svakog kraja spoja. S svakim sljedećim poslanim podacima, inicijalni brojevi se uvećavaju i tako primatelju omogućavaju ispravnu rekonstrukciju slijeda poruka. IL izbjegava slijepu retransmisiju. Ovo poboljšava učinkovitost u mrežama, gdje bi slijepa retransmisija dodatno zagušila mrežu. Poput TCP-a, IL ima prilagodiva vremena isteka *(eng. Timeout)*. Vrijeme potrebno paketu da prijeđe put od pošiljatelja do primatelja i natrag *(eng. Round-trip-time)* se koristi kako bi se dinamički odredila vremena potvrde podatka i ponovnog slanja.

IL protokol prednosti tok podataka između dvije krajnje točke. Način rada IL protokola može se prikazati strojem stanja vidljiv na slici 2.



ackok	any sequence number between id0 and next inclusive
!x	any value except x
–	any value

Slika 2. Dijagram stanja IL protokola [13]

Više o specifikacijama IL protokola i detaljnije objašnjenje dijagrama stanja se može naći u literaturi [13].

3. Programiranje

3.1 Programiranje u C-u

Korisnički programi u Plan 9 sustavu su napisani u više različitih jezika. Neki od njih su skripte napisane za školjku, neki su napisani u novom jeziku sličnom C-u, nazvanom Alef. Alef je napisan s ciljem boljeg paralelnog programiranja, no njegov razvoj je prekinut, ali su se izvukle pouke iz njega. Velika većina programa je napisana u dijalektu ANSI C [7] . Od njih, većina su potpuno novi programi, dok neki su nastali prije ANSI C specifikacije, ali su njoj prilagođeni.

Plan 9 dijalekt C-a ima neka manja unapređenja i nekoliko većih ograničenja. Prevodilac (*eng. Compiler*) zahtjeva da sve funkcije imaju ANSI prototipove i svi pozivi funkcija se pojavljuju u dosegu deklaracija prototipa. Deklaracije prototipova funkcija se spremaju i datoteku zaglavlja *(egn. header)* koja opisuje sve funkcije te biblioteke. Standardna biblioteka Plan 9 sustava je *libc,* tako da svi izvorni kodovi napisani u C-u za ovaj sustav moraju uključiti <libc.h> [3]. Drugo ograničenje je da C prevodioci primaju samo podskup pretprocesorskih naredbi. Glavni izostanak je *#if*, zbog česte nepotrebne upotrebe.

Veliki dio standardne biblioteke se poklapa s ANSI i POSIX specifikacijama jezika C, ali divergira kad se dođe do specifičnih načina implementacije sustava. U slučajevima kad se semantika standardne funkcije mijenja, onda se i ime same funkcije promijeni (npr. *Create* funkcija u Planu 9 obavlja sličnu funkcionalost kao i *creat* iz UNIX-a i prima dodatni parametar).

Još jedna razlika s obzirom na ANSI C je uporaba 16-bitnog skupa znakova nazvanog UNICODE, za kojeg je zaslužan Ken Thompson. Za Plan 9 su sve reprezentacije većih jezika uniformne kroz sve programe. Da bi se pojednostavnila razmjena teksta između programa, znakovi se pakiraju u bajtni tok podataka nazvan UTF-8, koji je priznat kao standard. Prednosti ovog pristupa su neovisnost od redoslijedu okteta *(end. Byte-order)*, i kompatibilnost s ASCII-jem.

Jedan dio programa ne slijedi navedene konvencije. Ti programi su preneseni od UNIX zajednice, poput *tex-a*. Da bi se izbjegla potreba za konvertiranjem svih

takvih programa s svakom novom verzijom, izgrađeno je prenosivo okruženje nazvano ANSI C/POSIX okruženje, ili APE. APE ima odvojene biblioteke i naredbe da bi se prenošenje omogućilo.

3.2 Prenosivost i prevođenje

Plan 9 radi na više platformi. Pošto je sustav distribuiran, moguće je izvršavati jedan dio izračunavanja na *SPARC* arhitekturi, drugi na *i386*, dok je je datotečni poslužitelj na MIPS-u. Da bi ova heterogenost bila transparentna, moraju postojati određene konvencije o izmjeni podatak između programa, a da bi održavanje programa bilo što jednostavnije, moraju postojati konvencije o inter-platformnom prevođenju *(eng.cross-architecture compilation).*

Da bi se izbjegli problemi s poretkom bajtova (*eng. Byte-ordering*), podatci između programa se izmjenjuju u tekstualnom obliku, kad god je to praktično. U slučajevima kad je potreban binarni format, podaci se šalju bajtnim tokom s predefiniranim načinom kodiranja za vrijednosti s više okteta.

Programi, kao i jezgra sustava često predstavljaju svoje podatke u obliku datoteka, kao sučelja koje je prenosivo. Tako se sistemski sat prezentira decimalnim brojem u */dev/time,* i *time* biblioteka čita tu datoteku (ne postoji sistemski *time* poziv). Još jedan primjer, bitniji za ovaj rad, je da jezgra sustava predstavlja stanje procesa u */proc/#pid,* kao datoteku *status.* Naredba *ps* je trivijalna – čita status datoteke zadanih procesa. Štoviše, nakon korištenja *import* naredbe, kao u opisano u Primjerima datotečnih poslužitelja, naredba *ps* ispisuje i Gunthorove procese.

Svaka podržana arhitektura ima vlastite prevodioce i punitelje koji stvaraju međukodne *(eng. Intermediate)* datoteke koje su prenosivo kodirane – sadržaj je jedinstven za ciljnu arhitekturu, ali je format neovisan o arhitekturi na kojoj je izvršena kompilacija [12]. Kad je prevodilac za danu arhitekturu preveden na drugoj arhitekturi, i onda korišten za prevođenje programa tamo, međukod stvoren na novoj arhitekturi je identičan međukodu stvorenom na matičnom procesoru. S pogleda prevodioca, svaka kompilacija je međuplatformna (*eng. Cross-compilation*). S strane punitelja za određenu arhitekturu je posve nebitno je li međukodna datoteka stvorena na ovoj ili nekoj drugoj arhitekturi.

Prevodilac i punitelj su za svaku arhitekturu posebno imenovani – ne postoji *cc* naredba. Na primjer, prevodilac za jezik C i arhitekturu Intel x86 se naziva *8c,* a punitelj *8l* (od x**8**6 arhitekture). Međukodne datoteke za ovaj primjer završavaju na .8, umjesto .o.

Plan 9 program za automatizirano kompiliranje se zove *mk* [10]*,* i vrlo je sličan onom s UNIX sustava *(make)*. On iz varijabli okruženja *(eng. Environment variables) $cputype* i *$objtype* određuje ciljnu arhitekturu. Ako se varijabla *$objtype* postavi na primjerice *SPARC*, onda se kompilacija obavlja za arhitekturu *SPARC*, bez obzira na *$cputype* koji označava arhitekturu na kojoj se obavlja prevođenje.

3.3 Paralelno programiranje

Plan 9 podupire paralelno programiranje s dva aspekta. Kao prvo, jezgra sustava pruža jednostavni model procesa i nekoliko pažljivo odabranih sistemskih poziva za sinkronizaciju i dijeljenje. Kao drugo, razvijen je novi jezik nazvan *Alef* koji podržava paralelno programiranje. Međutim, Rob Pike, jedan od autora *Alef-a* i Plan-a 9 je 2000.g objavio da se odustaje od daljnjeg razvoja tog jezika zbog previše složenog razvoja na više platformi. Pouke naučene na *Alef-u* su prenesene u biblioteke C-a (posebice dretvene, *eng.thread,* biblioteke) [3].

Moderni operativni sustavi pružaju dva mehanizma ostvarivanja paralelnog izvršavanja: Uobičajeni unixoidni procesi i jezgrene dretve. Početno se namjeravalo koristiti samo procese, te je njihova implementacija složenija i omogućava bolju kontrolu od procesa na UNIX-u. Jezgra sustava ima efikasno sučelje poziva sustavnih funkcija i jeftino stvaranje novih procesa i njihovo raspoređivanje. Uz procese se, prestankom razvoja jezika Alef i prenošenjem iskustava stečenih programiranjem u njemu te stvaranjem nove *thread* biblioteke, koriste i dretve. Više govora o *thread* biblioteci će biti kasnije, na pregledu praktičnog rada.

Paralelno programiranje treba ostvariti tri osnova zahtjeva: upravljanje resursima dijeljenima među procesima, sučelje prema raspoređivaču i fino-zrnata sinkronizacija procesa pomoću semafora *(eng. Spin lock)*. Novi procesi se stvaraju pomoću *rfork* poziva. *rfork* prima samo jedan argument, vektor bitova koji određuje koje od roditeljskih resursa će biti dijeljeni, kopirani, ili nanovo stvoreni za dijete.

Resursi pod kontrolom *rfork-a* uključuju imenički prostor, varijable okruženja, tablica opisnika datoteka, segmenti memorije i *note (eng. notes,* analogno UNIX signalima*).* Jedan od bitova upravlja da li će se stvarati novi proces ili će se modifikacije izvršiti nad procesom pozivateljem.

rendezvous sistemski poziv se koristi za sinkronizaciju. Prima dva parametra: *tag* i *vrijednost.* Kad jedan proces pozove *rendezvous,* on čeka dok se ne pojavi drugi proces s identičnim *tag-om* i onda oni razmjenjuju vrijednosti drugog parametra, *vrijednost.* Možda će nekome izgledati prilično primitivno, ali je dovoljno za obavljanje svih mehanizama međusobnog isključivanja.

4. Praktični dio rada

Ciljevi ovog rada su upoznavanje s konceptima, idejama, protokolima i načinom rada operacijskog sustava Plan 9. U tu svrhu sam napisao i mali program koji demonstrira načine stvaranja procesa, njihovu sinkronizaciju, i raspoređivanje unutar jezgre sustava. Da bi čitatelju pregled koda bio što jasniji, potrebno je obratiti pažnju na neke dodatne koncepte Plana 9, poput */proc* datotečnog sustava, dretvi *(eng. Thread)* i sl. Napravljeni program koji radi s procesima, u suštini stvara proizvoljan broj procesa, dio kojih stvara poruke, dok ih drugi čitaju. Komunikacija je ostvarena preko napravljene strukture koja je zaštićena sinkronizacijskim mehanizmima. Na kraju, svaki od procesa ispiše svoju statistiku u odabranu datoteku, ili na ekran. Uz taj program, napravljen je i program koji mjeri prosječno vrijeme potrebno da se ostvari komunikacija između dvije dretve preko predefinirane strukture *Channel.* Testiranje je pokazalo da je prosječno vrijeme oko devet mikrosekundi, uz jako mala odstupanja.

4.1 /proc datotečni sustav

Kao što je već naglašeno, Plan 9 kao jedan od svojih temeljnih koncepata ima i reprezentiranje resursa u obliku datoteka. /proc datotečni sustav je organiziran kao dvorazinska struktura direktorija [3]. Prva razina sadrži direktorije čija imena odgovaraju *pid-ovima* svih "živih" procesa. Unutar svakog direktorija su datoteke koje predstavljaju trenutno stanje procesa, ali služe i za kontrolu procesa. Neke od njih su:

/proc/n/args

/proc/n/ctl

/proc/n/fd

/proc/n/note

/proc/n/noteid

/proc/n/notepg

/proc/n/status

/proc/n/wait

Datoteka *args* sadrži argumente komandne linije (ukoliko je proces stvoren pozivom *fork(),* ili *rfork()*, onda je datoteka prazna), *fd* popis datoteka koje proces ima otvorene. Povratna vrijednost procesa operacijskom sustavu je proizvoljni niz znakova, a ne cijeli broj, kao što je to slučaj s UNIX-om. Tako datoteka *wait* sadrži povratne vrijednosti koje djeca ostavljaju svom roditelju kako bi on mogao saznati u kojem su stanju završila. Datoteka *ctl* je kontrolna datoteka. Upisom u nju može se dinamički mijenjati ponašanje procesa. Neke od zanimljivijih naredbi su:

stop	privremeno zaustavlja izvršavanje, proces prelazi u stanje *stopped*
start	nastavlja izvođenje procesa iz stanja *stopped*
close n	zatvara opisnik datoteke *n* (n se može pročitati iz *fd* datoteke)
kill	terminira proces sljedeći put kad prijeđe granicu korisnički / sustavni prostor
pri n	postavlja prioritet procesa na *n*

Prioritet procesa tumači raspoređivač procesa. Prioriteti se kreću u rasponu od 0 do 19, gdje veći broj označava viši prioritet. Proces ima osnovni i trenutni prioritet koji je manji ili jednak osnovnom. Kako proces sve više troši svoje dodijeljeno vrijeme, tako mu se trenutni prioritet smanjuje. Ukoliko se ne postavi drugačije, korisnički procesi imaju osnovni prioritet 10, dok sustavni imaju 13. Datoteka status se sastoji od teksta s dvanaest polja, kao što su: ime procesa, ime korisnika koji je pokrenuo taj proces, status procesa koji se dobije i naredbom *ps* , vremena izvršavanja procesa, količina zauzete memorije (osim stoga, u jedinicama od 1024 bajta), te osnovni i trenutni prioritet procesa. Plan 9 nema koncept signala koji se šalju procesima, kao što to ima UNIX, već koristi koncept asinkronih *note-a.* Datoteke *note, noteid,* i *notepg* služe da se *note* pošalje procesu, ili cijeloj procesorskoj grupi. Neke od primljenih *note-a* se mogu maskirati funkcijama koje će se izvršiti kad proces primi taj *note.*

4.2 rfork

rfork sistemski poziv je jedini način stvaranja novih procesa. Prima jedan argument koji određuje koji od roditeljskih resursa će biti dijeljen, kopiran, ili stvoren nanovo. Argument je dobiven logičkom operacijom *ili* između zastavica. Zastavice određuju: da li se stvara novi proces, ili se obavljaju modifikacije na pozivajućem procesu; da li dijete dijeli memoriju, opisnike datoteka i imenički prostor s roditeljem ili ne; da li se varijable okoline dijele s roditeljem ili se stvara novo prazno okruženje.

4.3 Dretve

Unutar biblioteke *thread.h,* nalaze se opisi funkcija za stvaranje novih dretvi, kao i procesa [3]. Dretve i procesi zauzimaju dijeljeni adresni prostor, unutar kojeg se sinkroniziraju i komuniciraju putem dijeljenih varijabli i kanala *(eng. channels).* Unutar jednog procesa može postojati više niti paralelnog izvođenja koje se nazivaju dretve. Dretve se raspoređuju bez prekidanja *(eng. Nonpreemptivley),* algoritmom *round-robin.* Što znači da se raspoređuju po principu *first-come-first-served.* Dretva mora eksplicitno prepustiti kontrolu procesora prije nego li se druga dretva tog procesa može nastaviti izvršavati. Ovo svojstvo se uvelike razlikuje od UNIX-ovog načina raspoređivanja dretvi gdje sustavski raspoređivač prekida izvođenje pojedine dretve i predaje kontrolu drugoj. Pozivi kojima dretva predaje kontrolu procesora su primarno vezani uz stvaranje novih dretvi i komunikaciju s drugim procesima ili dretvama. Uz te pozive, dretva može pozvati i sustavni poziv *yeild* kojim predaje kontrolu.

Struktura kanal-a *(eng. channel)* služi za razmjenu poruka između procesa i dretvi. To je spremnik za jednu ili više poruka *(eng. buffer)* proizvoljne duljine. Kad dretva ili proces šalje poruku *(eng. send)* preko kanala kojem se napuni spremnik (u slučaju da spremnika nema, tj. da je duljine jednake jedan), pozivatelj se blokira sve dok se ne dogodi odgovarajući poziv *(eng. recv)* za primanje te poruke.

Ovo svojstvo, uz nemogućnost prekidanja izvođenja dretvi dok one same ne predaju kontrolu je iskorišteno u izradi drugog programa koji mjeri vrijeme potrebno da se preko kanala pošalje poruka. Naime, uz glavnu se stvara još jedna dretva koja glavnoj preko kanala šalje pročitano vrijeme u nanosekundama. Nakon poziva *send* ta dretva se blokira i čeka odgovarajući primitak poruke. Glavna dretva primi poruku, i

zatim sama pročita vrijeme u nanosekundama. Razlika tih vremena je vrijeme potrebno da se preko kanala pošalje *vlong* tip podatka (uz ogradu da je to vrijeme uvećano za vrijeme potrebno glavnoj dretvi da očita sistemski sat). Proces slanja se ponavlja proizvoljan broj puta i na kraju se ispiše prosječno vrijeme. Bitno je napomenuti da, ako se ugase svi ostali nepotrebni programi na sustavu, nema većih odstupanja od već spomenutih 9 mikrosekundi (odstupanja su ranga do 1 mikrosekunde).

4.4 Iskustvo rada na sustavu

Plan 9 je javno dostupan na [21]. Instalacijski cd se može pokrenuti i u tzv. *"live"* načinu rada i tako omogućava isprobavanje sustava bez potrebe da se on instalira na tvrdi disk računala. Jezgra sustava je u usporedbi s jezgrom linux-a prilično ograničena po pitanju podržanih računalnih komponenti. Prilikom instalacije, promijenio sam četiri ili pet konfiguracija na kojima se Plan 9 odbijao instalirati. Napokon, instaliran je na računalu s 1.3 Ghz procesorom, 384MB SDRAM-a, 3dfx Voodoo 3 grafičkom karticom i tvrdim diskom kapaciteta 1.3 GB. Uzevši u obzir ova, u današnje vrijeme, velika ograničenja, performanse sustava dobivene testovima na opisanom računalu dobivaju novo značenje. Plan 9 ima vlastiti datotečni sustav, 9FS, kojeg ne prepoznaju drugi operativni sustavi. Neobična je činjenica da se, na moje prijenosno računalo, sustav instalirao bez problema i proradio koristeći program GRUB. Nažalost nije podržavao mrežnu karticu, te je rad na toj instalaciji bio ograničen.

Tijekom rada, pojavilo se i pitanje prenošenja koda s računala na kojem je napisan. Zadnja verzija Plana 9 je objavljena je 2002.g, i podržava starije verzije SSH protokola. Na lokalnoj mreži na koju je bilo spojeno računalo postoji FTP poslužitelj i naredbom *ftpfs* sam uspio prenijeti kod. Tu se pokazala i ljepota sustava, jer je FTP poslužitelj bio virtualno vidljiv u odabranom direktoriju, i prijenos podataka s poslužitelja i na poslužitelj je ostvaren korištenjem standardnih naredbi *cp, lc* (naredba gotovo identična *ls-u* na UNIX-u) i sl. Sustav nema internet preglednik pa je za traženje dokumentacije i objašnjenja na internetu potrebno koristiti dva računala. Olakotna okolnost je što je većina dokumentacije dostupne na službenim stranicama bila u obliku *PostScript* dokumenata dostupna i u */sys/doc* direktoriju.

Programsko razvojno okruženje *acme* [11] je jedno od ugodnih iznenađenja. Njegova zanimljivost je što na bilo kojem dijelu prozora, korisnik može napisati bilo koju naredbu izvršivu u školjci sustava (ili neku od naredbi svojstvenih *acme-u*), taj tekst označiti i pritiskom na srednju tipku (*button 2* u terminologiji Plana 9) miša izvršiti naredbu. Rezultat te naredbe je vidljiv u novostvorenom prozoru u stupcu u kojem je tekst napisan. Isto tako, desna tipka miša pretražuje tekst za pojavljivanje označene riječi. Navedeni primjeri samo su dio mogućnosti *acme-a*, više se može naći u literaturi.

Od napisanog koda o dretvama valja izdvojiti:

```
Alt alts[] = {
     /*   c          v              op    */
          {nil, &sender_time, CHANRCV},
          {nil, nil, CHANEND},
     };
alts[0].c = chancreate (sizeof (vlong), 0);
...
switch (alt(alts)){
               case 0:
                    my_time = nsec();
                    delta_time[i] = my_time - sender_time;
                    sum_of_delta +=delta_time[i];
                    printf ("delta_time[%d] = %d \n", i, (int)
delta_time[i]);
                    break;
               default:
                    sysfatal ("impossible");
          }
```

Navedeni odsječak prikazuje stvaranje kanala za komunikaciju između dretvi i primanje vremenske značke od druge dretve. Struktura *Alt* se sastoji od komunikacijskog kanala, varijable u koju se sprema primljena vrijednost i tip kanala. Funkcija *alt(struct Alt *)* radi slično kao funkcija *select()* na UNIX-u, te vraća indeks kanala koji je spreman za čitanje.

U programu s procesima, vrijedi pogledati izvedbu sinkronizacijskih mehanizama:

```
struct buff {
     QLock lock;
     Rendez is_full;
     Rendez is_empty;
     char data [512];
     int idx;
     int num_recv;
};
```

Struktura *buff* sadrži *Qlock* i *Rendez* varijable koje služe za sinkronizaciju. *Lock* varijable unutar strukture *Rendez* su postavljene tako da pokazuju na glavnu sinkronizacijsku varijablu, *Qlock lock.*

```
buff * create_buff (int num){
     buff * buffer;
     buffer = mallocz (sizeof *buffer, 1);
     buffer->is_full.l = &buffer->lock;
     buffer->is_empty.l = &buffer->lock;
     buffer->idx = 0;
     buffer->num_recv = num;
     return buffer;
}
```

Ukoliko je spremnik pun, a pošiljatelj pokušava poslati novi podatak, on će završiti u redu čekanja na varijabli *is_full*, sve dok ga neki od primatelja ne probudi. Pogledajmo funkciju za primanje poruke:

```
char * recv (buff * buffer){
     qlock (&buffer->lock);
     char * ptr = mallocz (strlen (buffer->data), 1);
     while (buffer->idx == 0 && buffer->num_recv >=0){
          printf ("going to rsleep\n");
          rsleep(&buffer->is_empty);
          printf ("woke up\n");
     }
```

```
        buffer->num_recv -=1;
        buffer->idx=0;
        ptr[0] = '\0';
        strcat (ptr, buffer->data);
        if (buffer->num_recv < 0) ptr = nil;
        rwakeup (&buffer->is_full);
        qunlock (&buffer->lock);
        return ptr;
}
```

Slično izgleda i funkcija za slanje poruke, te se čitatelju preporuča da je potraži na priloženom CD mediju.

5. Plan 9 u drugim sustavima

Zadnje, četvrto izdanje Plana 9 je objavljeno 2002.godine[1]. No, to nikako ne znači da Plan 9 i dalje nema utjecaja na računarski svijet. U ovom poglavlju ćemo ukratko pogledati neke od projekata na koje je Plan 9 direktno utjecao, te i dalje utječe.

5.1 Plan 9 from user space

Plan 9 je dobio ime po filmu "Plan 9 from outer space", čija radnja se može sumirati u jednoj rečenici: Izvanzemaljci su uskrsnuli mrtvace kao zombije i vampire u nadi da će spriječiti ostatak čovječanstva da napravi *Solarnite*, neku vrstu bombe. Vjerojatno pogađate da nije pretjerano dobro primljen od strane publike.

Plan 9 from user space je projekt koji kao cilj ima prenijeti većinu ideja, aplikacija i usluga na UNIX-oidne sustave [14]. Preneseni su programi poput *acme-a, sam-a, rc* školjke i sličnih. Personalizacija imeničnih prostora i potpuna implementacija 9P protokola nije direktno moguća. Umjesto toga, UNIX priključnice *(eng. Sockets)* služe kao posrednici prema *9pclient-u.* Svakako da implementacija nije toliko čista kao u Planu 9, ali obavlja posao. Paralelno programiranje je ostvareno pomoću biblioteke *thread,* a pozivi *rfork* nisu toliko moćni kao na nativnom Planu 9. Osobno sam instalirao ovaj projekt na svoje računalo i dojma sam da omogućava dobar pogled na Plan 9, uz sve programe dostupne na modernijim operativnim sustavima, poput internet preglednika.

5.2 Inferno

Inferno je kompaktni operacijski sustav osmišljen s namjenom izgradnje distribuiranih i mrežnih sustava[15]. Dostupan je javno na službenim stranicama.

Inferno se može izvršavati kao korisnička aplikacija na temelju gotovo svakog poznatijeg operacijskog sustava (od Windows-a, Mac-a, BSD-a, Linux-a, Solaris-a, Plan-a 9 i Irix-a), ili kao samostalan operacijski sustav. Nastao je na temeljima Plana 9, koristi ideje poput imeničkih prostora, JIT *(eng. Just in time)* prevođenja, mrežne raspodijeljenosti, prezentacije resursa poput datoteka, i standardnog

komunikacijskog protokola Styx. Styx je gotovo identičan 9P2000, zadnjoj verziji protokola 9P, objavljenoj s četvrtim izdanjem Plana 9. Inferno čak koristi i Plan 9 prevodioce. Više o ovom sustavu se može naći u literaturi [16].

5.3 Wmii

Wmii *(eng. window manager improved)* je "lagani", dinamički upravitelj prozorima koji radi s X11 poslužiteljem [17]. Prilagodljiv je *(eng. Scriptable)* i ima sučelje prema 9P datotečnom poslužitelju. Podržava tradicionalno i *acme-like* upravljanje prozorima. Koristio sam wmii i jako sam zadovoljan s njim. Uz wmii, razvijena je i biblioteka u ruby-ju nazvanu *libixp* [18] koja se koristi kao samostalan klijent/server za 9P protokol. Wmii je jako brz i pogodan za računala s starijim komponentama i za korisnike koji vole minimalistički pristup uz najveću učinkovitost.

5.4 V9fs

V9fs je omogućava 9P2000 klijent za jezgru linuxa 2.6 [19]. Izvorni kod se održava kao dio glavnog dijela *(eng. main-stream)* jezgre, od verzije 2.6. Dokumentacija ovog projekta se može pronaći pod glavnim stablom linux jezgre, pod *Documentation/filesystems/9p.txt.*

5.5 9P implementacije

9P kao protokol je jako učinkovit i postoji više implementacija 9P protokola u različitim programskim jezicima [20]. Više o tome možete pronaći u literaturi.

5.6 Glendix

Glendix je projekt ima za cilj izgraditi Plan9/Linux distribuciju [23]. Namjera je napraviti razvojno sučelje za programiranje distribuiranih sustava, kao što je to omogućeno u Planu 9, ali da radi na što većoj paleti platformi i komponenti. Više o temi i uspjesima prenošenja Plan 9 C biblioteke na linux i binarnom formatu za linux koji može koristiti Plan 9 punitelj *(eng. Loader)* se može pronaći u literaturi.

6. Zaključak

Cilj ovog rada je bilo upoznavanje s konceptima i načinima realizacije raspodijeljenog operacijskog sustava Plan 9. Plan 9 je zamišljen kao nasljednik UNIX-a. Danas, gotovo 20 godina od početka razvoja možemo reći da u toj zamisli nije bio uspješan. Međutim, nikako ne možemo reći da projekt nije uspješan. Plan 9 ima značajano drugačiji pristup operacijskim sustavima, pogotovo sad kad su ideje mrežne raspodijeljenosti sve više aktualne. Širokopojasni pristup prema korisnicima je realnost. Transparentnost je ključan aspekt sustava koji se želio ostvariti. U tu svrhu je osmišljen protokol 9P, koji se nameće kao najvažnija komponenta sustava. Ne samo kao protokol za pristup podacima, već kao i glavni koncept koji se primjenjuje u drugim sustavima. U prilog ovoj činjenici govori i Google summer of code iz 2007.god., gdje se radilo na implementaciji 9P protokola u PHP-u i Java scriptu [22].

7. Sažetak

Naslov: Istraživački sustav Plan 9

Sadržaj:

Plan 9 distribuirani operacijski sustav je temeljen na nekoliko pažljivo odabranih koncepata:

1. Sve je datoteka,
2. pristup datotekama se obavlja pomoću protokola 9P,
3. korisnici stvaraju svoje poglede na sustav.

Svaki uređaj se predstavlja korisniku u obliku datotečnog sustava. Upravljanje uslugama i uređajima se svodi na upisivanje vrijednosti u asocirane kontrolne datoteke, čime se ostvaruje jednostavno sučelje prema sustavu. Pristup datotekama se obavlja pomoću novog protokola 9P. 9P je okosnica sustava i ostvaruje nevjerojatnu transparentnost. Pomoću njega se može vrlo jednostavno (koristeći samo naredbu ili dvije) izvođenje lokalnog procesa prebaciti na CPU poslužitelj, ili pogledati procese koji se odvijaju na CPU poslužitelju. Najvažnije svojstvo je transparentnost prema korisniku. Korisnik stvara svoj vlastiti pogled na sustav tako okupljajući usluge koje su njemu bitne i izdvajajući one koje mu nisu od važnosti. To se radi u takozvanim imeničkim prostorima. Prelaskom na drugo računalo, korisnik samo ponovo izgradi svoj imenički prostor i prijelaz je gotovo neprimjetan.

Ključne riječi: Plan 9, Operacijski sustav, 9P, IL, imenički prostor, distribuirani operacijski sustav, transparentnost.

8. Summary

Title: Plan 9 reasrch system

Summary:

Plan 9 is a distributed operating system based on a few carefully chosen ideas:

1. Everything is a file,
2. access to files is done through 9P protocol, and
3. users create and modify their own namespace.

Every device is presented as a hierarchical file system. Controlling services and devices come down to writing values in associated control files, which is a base to achieving a simple system interface. File access is done using a new protocol, called 9P. 9P is a foundation of the system and achieves incredible transparency. With 9P, only one or two commands are sufficient for things like transporting process execution to a remote CPU server, or examining what remote CPU server is running. The most important feature of 9P is transparency to the user. User creates his own namespace, selecting and binding services and devices that are important to him, and discarding those which are of no immediate interest to him. Switching to another computer or terminal is very easy, one just has to reconstruct his private namespace.

Keywords: Plan 9, Operating system, 9P, IL, namespace, distributed operating system, transparency.

9. Literatura

[1] Plan 9 from Bell labs - *Overview* http://plan9.bell-labs.com/plan9/about.html 20. veljače 2008.

[2] Rob Pike, Dave Presotto, Sean Dorwald, Bob Flandrena, Ken Thompson, Howard Trickey, Phil Winterbottom, *Plan 9 from Bell Labs* http://plan9.bell-labs.com/sys/doc/9.html 20.veljače 2008.

[3] *Plan 9 programmer's manual* http://plan9.bell-labs.com/sys/man 6. ožujka 2008.

[4] Rob Pike, Dave Presotto, Ken Thompson, Howard Trickey, Phil Winterbottom The use of namespaces in Plan 9 - *The use of namespaces in Plan 9* http://plan9.bell-labs.com/sys/doc/names.html 10. ožujka 2008.

[5] Dave Pressoto, Phil Winterbottom The organization of Networks in Plan 9 – *The organization of Networks in Plan 9* http://plan9.bell-labs.com/sys/doc/net.html 20.ožujka 2008.

[6] Tom Duff, Rc – The Plan 9 Shell - *Rc - The Plan 9 Shell* *http://plan9.bell-labs.com/sys/doc/rc.html* 4.ožujka 2008.

[7] Rob Pike, How to Use the Plan 9 Compiler - *How to Use the Plan 9 Compiler* http://plan9.bell-labs.com/sys/doc/comp.html 8.ožujka 2008.

[8] Rob Pike, 8½, the Plan 9 Window System - *8½, the Plan 9 Window System* http://plan9.bell-labs.com/sys/doc/8%C2%BD.html 25.ožujka 2008.

[9] Rob Pike, The text editor *sam* - *The text editor sam* http://plan9.bell-labs.com/sys/doc/sam.html 25.travnja 2008.

[10] Bob Flanderna Plan 9 MkFiles - *Plan 9 MkFiles* http://plan9.bell-labs.com/sys/doc/mkfiles.html 25.travnja 2008.

[11] Rob Pike, Acme: User interface for Programmers - *Acme: User interface for Programmers* http://plan9.bell-labs.com/sys/doc/acme.html 8.ožujka 2008.

[12] Ken Thompson, Plan 9 C Compilers - *Plan 9 C Compilers* http://plan9.bell-labs.com/sys/doc/compiler.html 10.ožujka 2008

[12] Dave Pressoto, Phil Winterbottom, The IL protocol - *The IL protocol* http://plan9.bell-labs.com/sys/doc/il.html 10.travnja 2008.

[13] Rob Pike, Dave Pressoto, Ken Thompson, Gerard Holzmann, Process Sleep and Wakeup on Shared-memory Multiprocessor - *Process Sleep and Wakeup on Shared-memory Multiprocessor* http://plan9.bell-labs.com/sys/doc/sleep.html 25. ožujka 2008.

[14] Plan 9 from user space - *Plan 9 from user space* http://swtch.com/plan9port/ 27.travnja 2008.

[15] Inferno - *A compact operating system for building cross-platform distributed systems* http://www.vitanuova.com/inferno/ 23. ožujka 2008.

[16] Inferno – *Documentation* http://www.vitanuova.com/inferno/docs.html 23.ožujka 2008.

[17] Wmii – Window manager improved 2 http://www.suckless.org/wiki/wmii/ 12. srpnja 2007.

[17] 9P FAQ http://9p.cat-v.org/faq 15.svibnja 2008.

[18] Libxp http://www.suckless.org/wiki/libs/libixp 28.listopada 2007.

[19] v9fs http://swik.net/v9fs 15.svibnja 2008.

[20] 9P implementations http://9p.cat-v.org/implementations 2.svibnja 2008.

[21] Plan 9 download http://plan9.bell-labs.com/plan9/download.html 23. veljače 2008.

[22] Plan 9 and Inferno at Google summer of code 07 http://gsoc.cat-v.org/about 14.svibanj 2008.

[23] Glendix: *Introduction* http://glendix.org/ 12.veljače 2008, 25.svibnja 2008.