Ulazno-izlazni podsistem

Operativni sistem treba da ima podsistem koji će obezbediti komunikaciju sa ulazno-izlaznim uređajima. Većina uređaja koji se priključuju na računar može se svrstati u neku opštu kategoriju. Opšte kategorije su formirane prema nameni uređaja, tako da se mogu izdvojiti uređaji za dugotrajno skladištenje podataka (*diskovi i trake*), uređaji za prenos podataka (*mrežne kartice i modemi*) i uređaji koji obezbeđuju interfejs ka korisniku (*monitori, tastature i miševi*).

Operativni sistem mora da obezbedi podršku za rad sa širokim spektrom ulazno-izlaznih uređaja čije karakteristike variraju kako sa vrstom uređaja, tako i s konkretnim modelom određenog proizvođača hardvera. Ulazno – izlazni podsistem je deo operativnog sistema koji obavloja sledeće funkcije:

1. upravlja ulazno-izlaznim uređajima i operacijama koje ti uređaji izvršavaju i kontroliše ih
2. obezbeđuje što jednostavniji interfejs prema korisniku i ostatku sistema

Zbog velikih razlika u samim uređajima, potrebno je u sistem implementirati različite metode kontrole uređaja, a sama kompleksnost upravljanja uređajima mora biti sadržana i sakrivena u jezgru. U savremenim tehnologijama zastupljena su dva konfliktna trenda: standardizacija softverskih i hardvesrkih interfejsa i povećanje broja ulazno-izlaznih uređaja različitih karakteristika. Kao rezultat, podrška za konkretne uređaje ili grupe srodnih uređaja sadržana je u programima za upravljanje ulazno-izlaznim uređajima – tj drajverima (*drivers*). Drajveri predstavljaju unikatni interfejs ulazno-izlaznog podsistema kao što sistemski pozivi predstavljaju univerzalni interfejs između aplikacija i operativnog sistema.

Klasifikacija uređaja

Ulazno-izlazni uređaji razlikuju se u mnogim aspektima. Karakteristike zavise od vrste uređaja i od konkretnog modela određenog proizvođača hardvera. Klasifikovaćemo uređaje prema sledećim kriterijumima:

1. ***namena uređaja***
   Prema nameni, uređaji se mogu svrstati u sledeće opšte kategorije:
   - uređaji za dugotrajno skladištenje podataka (*storage devices*) tj memorijski medijumi velikog kapaciteta, kao što su diskovi i trake
   - uređaji za prenos podataka (*transmission devices*), kao što su mrežne kartice i modemi
   - uređaji koji obezbeđuju interfejs ka korisniku (*human-interface devices*), kao što su monitori, tastature i miševi
2. ***smer prenosa***
   Prema smeru prenosa, uređaji se mogu podeliti na ulazne (miš, skener), izlazne (štampač) i ulazno-izlazne (mrežna kartica)
3. ***jedinična količina prenetih podataka***
   Prema jediničnoj količini prenetih podataka, tj minimalnoj količini podataka za čitanje i upis, uređaji se dele na one koji rade sa ***blokovima (block devices)*** i one koji rade sa ***znakovima (character devices***). Kao specijalna klasa mogu se izdvojiti mrežni uređaji.
   Za uređaje koji rade sa blokovima, jedinična količina prenetih poodataka jeste blok podataka. Interfejs za blok uređaje sadrži sve elemente potrebne za pristupanje diskovima i drugim blok uređajima.
   Za uređaje koji rade sa znakovima, jedinična kolličina prenetih podataka jeste bajt.
4. ***metoda pristupa***
   Po metodi pristupa, uređaji se dele na uređaje sa sekvencijalnim pristupom (kao što su modem i magnetna traka) i uređaje sa direktnim ili slučajnim pristupom (kao što su diskovi i CD-ROM uređaji). Sekvencijalni uređaji prenose podatke fiksnim redosledom koji određuje sam uređaj, dok je kod uređaja sa direktnim pristupom redosled prenosa podatka proizvoljan i određuje ga aplikacija tj korisnik.

5. ***deljivost uređaja***
   Prema deljivosti, uređaji se mogu klasifikovati kao deljivi i nedeljivi. Deljive uređaje može korisiti veći broj procesa istovremeno. Deljivi uređaji su, na primer, diskovi (veći broj korisnika prijavljenih na udaljeni UNIX server istovremeno koristi disk servera preko sistema datoteka). Nedeljive, tj posvećene uređaje, karakteriše ih to što u jednom trenutku može da ih koristi samo jedan proces. Primer nedeljivog uređaja je tastatura
6. ***brzina uređaja***
   Prema brzini, uređaji se mogu grubo podeliti na brze i spore. Ova podela je relativna jer brzina varira od nekoliko bajtova u sekundi do više gigabajta u sekundi (mrežna kartica).
7. ***mogućnost upisa***
   Prema mogućnosti upisa, uređaji se dele na one koji služe za čitanje i pisanje (kao što su diskovi), one koji isključivo služe za čitanje (CD-ROM) i one koji služe isključivo za pisanje (grafička kartica).

Hardver značajan za ulazno-izlazni podsistem

Uređaji komuniciraju s računarskim sistemom tako što šalju signale preko žičnih ili bežičnih prenosnih medijuma, a na računar su vezani preko odgovarajućih priključaka

Magistrale

Ukoliko više uređaja deli zajednički skup žica, sa strogo definisanim protokolom koji specificira skup poruka što se mogu poslati, veza s računarom se naziva ***magistrala (bus)***. Serijska veza uređaja kod koje je samo jedan uređaj priključen na sistem, a svaki uređaj ima priključak za povezivanje sledećeg, naziva se lanac (*daisy - chain*) i funkcioniše slično magistrali.

Sistemska magistrala PCI spaja procesorsko-memorijski podsistem sa brzim ulazno-izlaznim uređajima. Sprecijalna magistrala koja se naziva magistrala za proširenje (*expansion bus*) sadrži i serijske i paralelne priključke (*ports*) na koje se priključuju spori uređaji kao što su tastatura, miš, štamapač.

Kontroleri

Na slici su prikazana dva SCSI diska priključena na SCSI kontroler. Kontroler radi na tri načina: kao priključak (*port*), kao magistrala (*bus*) ili kao uređaj (*device*). Na primer, serijski kontroler je priključak. Grafička kartica je kontroler koji se priključuje na PCI magistralu, a radi kao uređaj. SCSI kontroler se s jedne strane priključuje na PCI magistralu, a s druge strane formira novu magistralu (SCSI) na koju može da se poveže do 15 SCSI uređaja.

Svaki kontroler ima jedan ili više registara. Upisom i čitanjem vrednosti ovih registara obavlja se komunikacija između procesora i kontrolera, tako da procesor može izdati komande kontroleru kako bi se npr, obavio prenos podataka na izlazni uređaj.

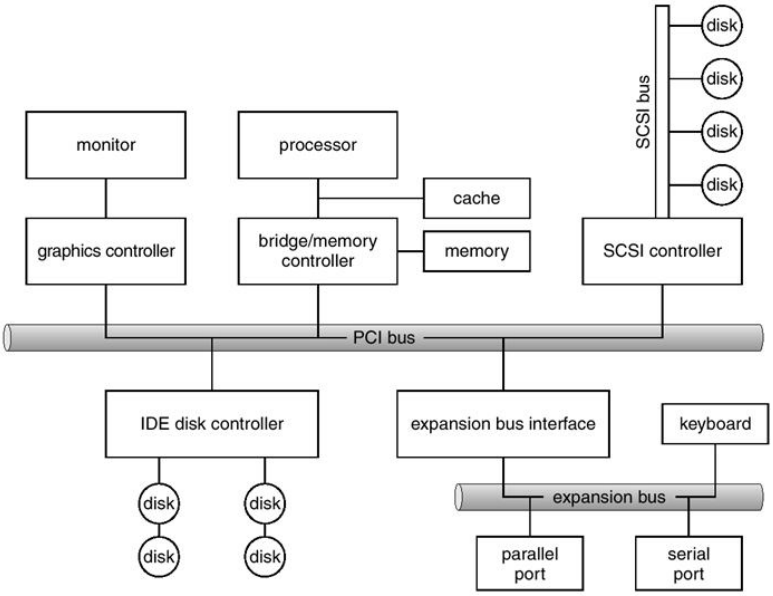
Registrima kontrolera može se pristupiti na dva načina, zavisno od toga da li su memorijski i ulazno-izlazni prostor razdvojeni. Nekim uređajima se može pristupiti na oba načina, zavisno od operacije koju treba obaviti. Tipičan primer je grafička kartica.

Tipičan ulazno-izlazni priključak se sastoji od četiri registra:

1. ***kontrolni registar (CONTROL)***
   Služi za postavljanje režima rada uređaja, kao što su brzina priključka i tip komunikacije. U ovaj registar, procesor isključivo upisuje podatke. Namenjen je direktnom upravljanju hardverom samog I/O uređaja. Kod njega svaki bit ima neku funkciju u upravljanju radom I/O uređaja

2. ***statusni registar (STATUS)***
   Ovaj registar opisuje status komande koja se izvršava, tj da li je komanda izvršena, da li je podatak spreman, da li je nastupila greška itd. Procesor iz statusnog registra isključivo čita podatke. Služi da preko njega procesor PC-a dobije informacije o stanju u kojem se nalazi I/O uređaj. Npr štampač, kao jedan I/O uređaj , preko svog statusnog registra, može obavestiti procesor da li je spreman da prihvati podatke za štampu, da li mu je potrebno papira, itd.
3. ***registar podataka za ulazni režim (DATA-IN)***
   Služi za prihvat podataka koji pristižu sa sistemske magistrale (najčešće od strane procesora) i koji treba da se obradi u I/O uređaju.

4. ***registar podataka za izlazni režim (DATA-OUT)***
   Služi da se preko njega na sistemsku magistralu postavi podatak koji potiče od I/O uređaja i koji će biti prihvaćen od strane procesora, DMA kontrolera ili drugog I/O uređaja.



Princip povezivanja uređaja (PC arhitektura)

Upisivanjem i čitanjem podataka sa odgovarajućih priključaka, procesor obavlja ulazno-izlazne operacije i upravlja radom uređaja.

Tehnika prozivanja

Računar i kontroler funkcionišu po principu proizvođač-potrošač, koji se u ovom slučaju realizuje pomoću dva bita: ***busy i command ready.***

- *Busy* bit u statusnom registru opisuje stanje kontrolera. Ukoliko je bit postavljen, kontroler je zauzet, tj nešto radi. Ako bit nije postavljen, kontroler je slobodan i spreman da prihvati novu komandu.
- Računar ukazuje kontroleru na prisustvo nove komande preko *command-ready* bita u komandnom registru. Kada računar postavi ovaj bit, kontroler je dobio komandu i izvršavanje može da počne.

Tehniku sporazumevanja (*handshake*) objasnićemo na primeru slanja jednog bajta na serijski kanal, koje se odvija u sledećim koracima:

1. računar čita statusni registar. Čitanje se ponavlja sve dok vrednost *busy* bita ne bude 0
2. računar postavlja *write* bit u komandnom registru i upisuje bajt za serijski kanal u *data-out* registar
3. računar postavlja *command-ready* bit
4. postavljen *command-ready* bit je znak kontroleru da treba da otpočne izvršenje komande. Kontroler postavlja *busy* bit.
5. kontroler čita svoj komandni registar, detektuje komandu **write,** zatim čita *data-out* registar i organizuje ciklus za slanje sadržaja *data-out* registra na serijski kanal
6. ukoliko je sve u redu, kontoler briše *command-ready* bit, zatim u statusnom registru briše *error* bit koji ukazuje na pojavu greške, i briše *busy* bit koji ukazuje na to da je komanda završena. Počev od kraja koraka 4, računar stalno proverava da li je *busy* bit 0, kako bi znao kada je komanda završena.

Ovakva petlja mora dase ponovi za svaki ciklus. U koracima 1 i 6 imamo tehniku prozivanja (*polling*), koja se takođe naziva „zauzet čekanjem“ (*busy waiting*). Računar ponavlja čitanje statusnog registra i analizira *busy* bit sve dok vrednost ne postane 0. Za brze uređaje ovo traje relativno kratko, za spore veoma dugo.
Tehnika prozivanja se na mnogim procesorima obavlja u petlji sastavljenoj od tri instrukcije:

- čitanje statusnog registra
- ekstrakcija i analiza *busy* bita
- povratak u petlju ako je vrednost *busy* bita 1, izlaz iz petlje ako je vrednost *busy* bita 0

U slučaju sporih uređaja, procesor ostaje zauzet čekanjem u kranje dugačkim petljama, iako bi mogao da radi pametnije stvari. Međutim, za sada je *busy* bit jedini način da računar odredi da li je kontroler slobodan ili da li je komanda izvršena.

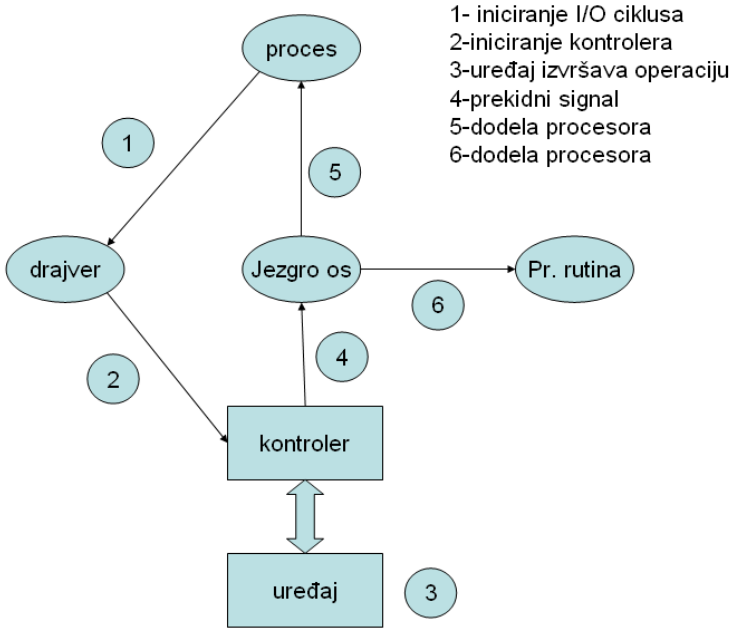
Prekidi

Osnovni nedostatak tehnike prozivanja je trošenje procesorskog vremena na petlju prozivanja. Nedostatak se može ukloniti uvođenjem hardveskog mehanizma koji omogućava uređaju da označi procesoru kada je komanda izvršena. Ovaj mehanizam se naziva ***prekid (interrupt)***.

Prekidni sistem funkcioniše na sledeći način:

- kada uređaj završi operaciju, kontroler šalje prekidni signal procesoru preko prekidne linije (*interrupt request line, IRQ*).
- Procesor će završiti tekuću instrukciju, posle čega se prekida sekvencijalno izvršavanje programa, kako bi se obradio prekid
- Posle obrade prekida, procesor nastavlja sa izvršavanjem programa

U princip, prekinuti proces mora da nastavi izvršavanje (da li odmah nakon obrade prekida ili neki drugi put – to zavisi od raspoređivanja procesa). Prekinuti proces može nastaviti izvršenje ukoliko se pre obrade prekida sačuva kontekst procesa, vrednost programskog brojača (*program counter PC*), registara procesora i pokazivača na deo memorije koji proces koristi. Kada se sačuva stanje procesa, poziva se rutina za obradu prekida (*interrupt handler*), koja obrađuje prekid i opslužuje prekid. Posle toga se može nastaviti izvršavanje prekinutog procesa.



Obrada prekida

Obrad prekida je data na slici:

- Proces najpre inicira ulazno-izlazni ciklus koji se prosleđuje drajveru – korak 1
- Drajver inicira kontroler – korak 2
- Uređaj zatim izvršava operaciju – korak 3
- Nakon završetka operacije kontroler postavlja prekidni signal – korak 4
- Prekidni signal prekida izvršavanje tekućeg procesa i predaje kontrolu rutini za obradu prekida – korak 5
- Nakon izvršene prekidne rutine, obnavlja se kontekst prekinutog procesa i kontrola predaje prekinutom procesu – korak 6

U modernim operativnim sistemima, tehnika prekida uključuje brojne sofisticirane osobine:

- Mogućnost odlaganja (*deffer*) obrade prekida dok je proces u kritičnoj sekciji
- Brzu i efikasnu tehniku određivanja uređaja koji je postavio prekid
- Omogućeno je gnežđenje prekida (*interrupt nesting*) i višeslojni prekidni sistem, koji će razlikovati prioritet prekidnih signala i rešavati prekide po prioritetu

Sve ove kvalitetne funkcije realizuje hardveski uređaj koji se naziva prioritetni prekidni kontroler (*priority interrupt controller PIC*). Većina procesora umesto jedne ima dve posebne linije za prekidne signale

1. nemaskirajuća (*non-masking*)
2. maskirajuća (*masking*) linija

Prekidni signal poslat po nemaskirajućoj liniji uvek može da prekine izvršenje tekućeg procesa. Nemaskirajuća linija se koristi za slanje prekidnih signala zbog kritičnih hardverskih grešaka, kao što su greške u memoriji.

Prekidni signali poslati po maskirajućoj liniji ne prekidaju izvršenje procesa sve dok se proces, npr nalazi u kritičnoj sekciji. Maskirajuća linija se koristi za slanje prekidnih signala koji potiču od normalnih operacija i standardnih grešaka koje se javljaju na I/O uređajima.

Prekidni sistem funkcioniše na sledeći način:

- relativno mali broj uređaja može da postavi prekidni signal, a za svaki od njih postoji posebna prekidna rutina, koja se nalazi u memoriji
- informacije o adresama svih prekidnih rutina čuvaju se u tabeli prekidnih vektora (*interrupt vector table*), gde svaki ulaz sadrži adresu jedne prekidne rutine
- na ovaj način se smanjuje trajanje određivanja uređaja koji je postavio prekidni signal
- tačno se zna koja je klasa uređaja vezana za koju liniju PIC kontrolera, tako da na osnovu aktivne linije, PIC okida procesor i daje mu informaciju u vidu vektora, tj pokazivača na ulaz u tabelu prekidnih vektora
- na ovaj način se izbegava prozivanje uređaja, a sve informacije obezbeđuje PIC

Prekidni sistem, po pravilu, primenjuje prekidne prioritetne nivoe, a to je mehanizam koji omogućava da se odloži opsluživanje prekida niskih prioriteta, dok se opslužuju prekidi visokih prioriteta – naravno bez maskiranja. Takođe, kad naiđe prekid višeg prioriteta, prekida se opsluživanje prekida nižeg prioriteta, pa se opslužuje prekid višeg prioriteta (pretpražnjenje - *preemption*).

U modernim OS, prekidi su brojni. Tokom podizanja sistema (*booting*), ispitivanjem magistrale se određuje da li su uređaji prisutni i ako jesu postavlja se odgovarajuća prekidna rutina u tabeli prekidnih vektora. Za vreme ulazno-izlaznih ciklusa, uređaj postavlja prekidni signal kad god zahteva servisiranje tj kada je komanda gotova ili kad se javila greška. Prekidni mehanizam se koristi i za rukovanje izuzecima (*exceptions*), kao što su PF greška, deljenje nulom, pristup zaštićenim adresama i greška u parnosti. U svim slučajevima, prekida se izvršavanje tekućeg procesa, a zatim se na bazi vektora iz tabele prekidnih vektora čita adresa odgovarajuće servisne rutine.

DMA (Direct memory access)

Kada uređaj pripremi podatke (jedan ili više bajtova), procesor treba da preuzme svaki bajt iz kontrolera. Pri tome, procesor svaki put proverava statusni bit, što znači da se izvršavaju dva U/I ciklusa. Ova tehnika čitanja podataka se naziva ***programirani ulaz-izlaz (PIO).***

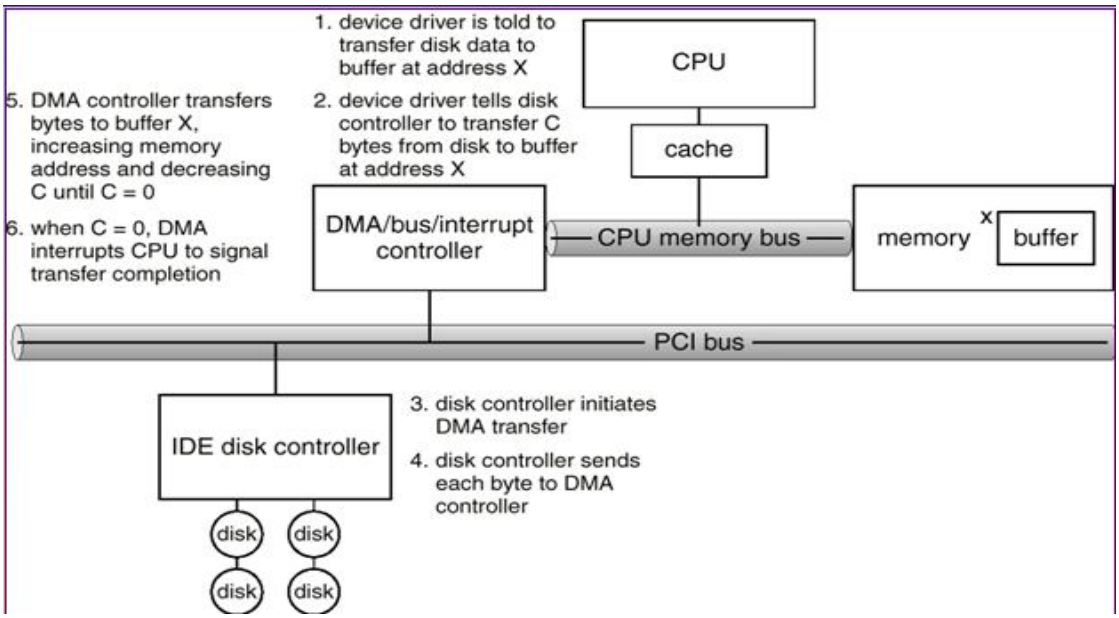
Kada se preko uređaja kao što su diskovi prenose ogromne količine podataka, PIO tehnika unosi veliki gubitak vremena. Kako bi se prenos podataka mogao obaviti na relaciji kontroler – radna memorija i obrnuto, specijalno je projektovan uređaj koji je naziva DMA kontroler. DMA (*direct memory access*) ima signale za

upravljanje transferom sa kontrolera, a ima i signale kojima upravlja memorijskim ciklusima. Da bi se inicirao DMA transfer, DMA se mora programirati, tj mora se definisati početna memorijska adresa, blok bajtova koje treba preneti i smer prenosa. Za PC arhitekturu,matična ploča sadrži više DMA kanala i svaki ima svoj U/I kontroler.

Objasnićemo jednostavan prenos podataka na relaciji disk – memorija preko DMA kontrolera. Transfer se obavlja u sledećim koracima:

- proces nalaže drajveru da penese podatke sa diska u bafer koji se nalazi memorijskoj adresi X
- drajver nalaže kontroleru diska da prenese C bajtova sa diska u bafer na memorijskoj adresi X
- kontroler diska inicira DMA transfer
- DMA kontroler prenosi podatke u bafer, uvećavajući memorijsku adresu X i smanjujući broj bajtova C nakon svakog prenesenog bajta
- Kada je penesen poslednji bajt (C=0), DMA šalje prekidni signal procesoru, obaveštavajući ga o završenom transferu

Protokol usaglašavanja između DMA i U/I kontrolera odvija se preko para linija, signalima ***DMA-request i DMA-acknowledge***. U/I kontroler inicira DMA ciklus samo kada ima spreman podatak, postavljanjem signala ***DMA-request.*** Posle primanja ovog signala, DMA zauzima memorijsku magistralu, a zatim organizuje memorijski ciklus u kome se bajt iz U/I kontrolera prenosi u memoriju. Po okončanju ovog ciklusa, DMA kontroler postavlja signal ***DMA-acknowledge,*** što znači kraj jednog DMA ciklusa. DMA automatski inkrementira adresu memorijskog bafera za sledeći DMA ciklus, a smanjuje boje batova za prenos. DMA postavlja prekidni signal procesoru kada prenese bajtove.
Kada je memorijska magistrala zajednička za CPU i DMA, ona u jednom trenutku može pripadati samo jednom od tih resursa. Postoje hardverski signali za njihovu sinhronizaciju, pri čemu procesor može svoje zahteve zadovoljiti iz keš memorije. Pri tome se na relaciji procesor – DMA dešava krađa ciklusa: DMA čeka da magistrala bude slobodna i zauzima je, a za to vreme procesor mora da čeka ukoliko želi memoriju i tako gubi cikluse. Bez obzira na ovu krađu, DMA potpuno rasterećuje procesor od transfera na relaciji memorija – uređaj, čime se osetno poboljšavaju performanse sistema.



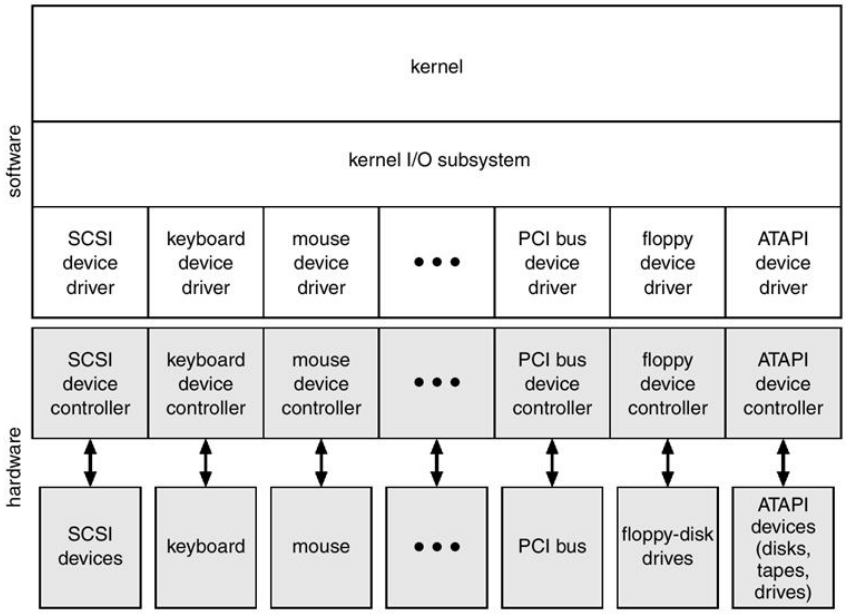
DMA kontroler

Uniformi interfejs ka aplikacijama

Jedan od ciljeva koje treba postići pri projektovanju operativnog sistema jeste formiranje uniformnog interfejsa ka aplikacijama i korisniku. Definisaćemo strukture i interfejse operativnog koji omogućavaju tretiranje ulazno-izlaznih uređaja na standardni, jedinstven način. Problem uniformnosti se rešava metodama apstrakcije i višeslojnosti softvera u okviru ulazno-izlaznog podsistema.

Srodni uređaji, kao što su diskovi, grupišu se u opšte klase uređaja. Svakoj klasi uređaja pristupa se preko standardnog skupa funkcija, koja se naziva ***interfejs***. Dakle, interfejs je standardni skup funkcija preko kojih se obraćamo istoj klasi ulazno-izlaznih uređaja. Razlike koje postoje među uređajima iste klase sakrivene su u specijalnim modulima jezgra operativnog sistema koji se nazivaju upravljčki programi ili drajveri (*device drivers*). Drajveri postoje za svaki uređaj, ali se korisniku predstavljaju kao jedan univerzalni interfejs.

Iznad svih drajvera nalazi se sloj ulazno-izlaznog podsistema jezgra koji je praktično nezavisan od hardvera – svi detalji vezani za konkretan hardver prepušteni su drajverima. Proizvođači uređaja, po pravilu, za svoje uređaje pišu drajvere za razne operativne sisteme, tako da se njihovi uređaji mogu priključiti bez intervencije projektanata samog operativnog sistema. Za proizvođače uređaja nezgodno je to što za svaki operativni sistem moraju da napišu poseban drajver za konkretni uređaj.



Apstrakcija uređaja i drajveri

Dakle, srodni uređaji su grupisani u klase, a drajveri apstrahuju karakteristike konkretnih uređaja. Programeru se dalje nudi standardni skup sistemskih poziva za rad sa ulazno-izlaznim podsistemom, kao što su: ***pozivi za rad sa memorijski mapiranim datotekama, pozivi za mrežne funkcije.*** Deo jezgra zadužen za rad s ulazno-izlaznim uređajima koordinira širok spektar usluga koje su na raspolaganju aplikacijama i ostalim delovima jezgra, kao što su:

- Upravljanje imenima datoteka i uređaja (*name space*)
- Kontrola pristupa datotekama i uređajima
- Kontrola operacija
- Dodela uređaja procesima na korišćenje
- Raspoređivanje ulazno-izlaznih operacija
- Baferovanje, keširanje, spuler
- Kontrola statusa uređaja
- Konfiguracija i iniciranje drajvera

Pri tome, jezgro operativnog sistema čuva informacije o stanju uređaja pomoću različitih struktura podataka smeštenih u delu memorije koji pripada jezgru. Tipične strukture koje opisuju stanje uređaja jesu tabele otvorenih datoteka i strukture koje čuvaju informacije o mrežnim konekcijama.

Usluge koje obezbeđuje ulazno-izlazni podsistem

Jezgro operativnog sistema obezbeđuje više usluga koje se odnose na ulazno-izlazne operacije. U značajne usluge spadaju raspoređivanje ulazno-izlaznih operacija, baferovanje, keširanje, spuler i upravljanje greškama.

Baferovanje

Bafer je deo memorije koji funkcioniše na principu proizvođač-potrošač i služi za čuvanje privremenih podataka prilikom prenosa podataka između dva uređaja ili između uređaja i aplikacije. Baferovanje se izvodi iz sledećih razloga:

1. usklađivanje različitih brzina između potrošača i proizvođača
2. prilagođavanje različitih veličina transfera podataka
3. održavanje semantike kopiranja

Zbog velike razlike u brzini, ulazno-izlazne operacije se mogu preklopiti, ali se tada bafer mora podeliti na dva dela, koje alternativno koriste proizvođač i potrošač. Takav sistem se naziva dvostruko baferovanje. Dvostruko baferovanje je jedna od tehnika koja se koristi pri keširanju čvrstih diskova. Smisao je u sledećem: postoje dva bafera (TRANSMIT i RECEIVE) koji se koriste za befrovanje podataka na relaciji memorija – disk odnosno disk – memorija, čime se premošćava razlika u brzini između memorije i diska. Tehnika dvostrukog baferovanja je recimo implementirana u Microsoft SmartDrive.

Baferi se veoma često koriste u mrežnim okruženjima, gde se velike poruke pre slanja dele na manje pakete. Paketi se primaju u bafer po nedeterminističkom redosledu (što zavisi od pojave grešaka na mreži, sukoba okvira itd), nakon čega se iz bafera rekonstruiše početna poruka.

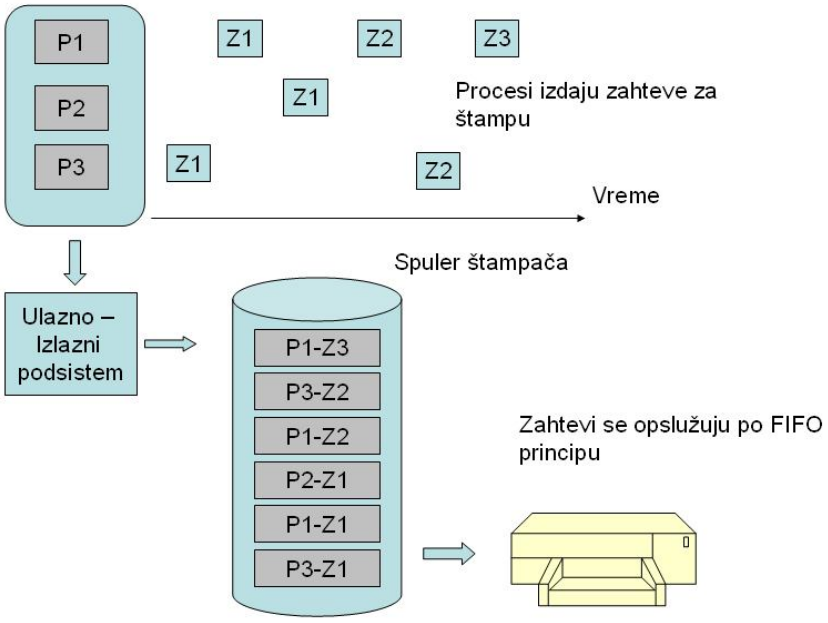
Održavanje semantike kopiranja ćemo objasniti na sledećem primeru: korisnički proces izdaje sistemski poziv za upis, tj slanje podataka na neki uređaj. Ukoliko je poziv neblokirajući ali asinhroni, proces dobija mogućnost da u toku slanja izmeni podatke koje želi da pošalje na uređaj. To znači da proces može upisati nešto drugo, a ne ono što je inicijlano zadato. Da bi se to izbeglo, tj da bi se očuvala semantika kopiranja, operativni sistem u svom jezgru definiše bafere u koje kopira korisnički bafer, pa tek onda predaje kontrolu korisničkom procesu.

Keširanje

Keš predstavlja oblast brze sistemske memorije koja čuva kopiju podataka, obično sa diska. Pristup podacima u kešu znatno je brži od pristupa podacima na ulazno-izlaznim uređajima. Keširanje predstavlja tehniku kopiranja delova diska u keš memoriju, čime se osetno poboljšavaju performanse U/I sistema diska.

Najjednostavnije rečeno, razlika između keša i bafera sastoji se u tome što bafer čuva trenutno aktuelne podatke, a keš čuva bilo koju kopiju da diska. Isti memorijski prostor može se koristiti i za baferovanje i za keširanje. Po pravilu, keširanjeje vrlo značajno i realizuje se u više nivoa, pri čemu se koriste različite tehnike za popunu i zamenjivanje podataka u kešu.

Spuler



Spuler štampača

Spuler je bafer koji privremeno čuva izlazne podatke namenjene nekom nedeljivom uređaju (npr štampaču). Spuler omogućava istovremeni pristup nedeljivim uređajima naslednećі način: procesi upisuju podatke namenjene uređaju na disk, a operativni sistem upravlja spulerom tako što opslužuje jedan po jedan zahtev. Na primer, svaki proces koji želi nesto da pošalje na štampu, ostavlja svoj zahtev u spuler na disku. Proces koji radi u pozadini upravlja spulerom i štampa jedan po jedan zahtev. Korišćenje spulera ima svoje prednosti:

- proces relativno brzo postavlja svoj zahtev u bafer, a nakon toga je slobodan da dalje obavlja svoje aktivnosti
- nedeljivi uređaji se koriste kao prividno deljivi, što omogućava većem broju procesa da istovremeno koriste uređaj

Kada se ne bi koristio spuler, procesi bi morali najpre da čekaju da se uređaj oslobodi kako bi ga mogli koristiti, a zatim bi se ponovo blokirali i čekali da uređaj završi operaciju koju su inicirali.