Operacijski sustavi

Svaki uređaj kojim upravlja neki mikroprocesor nužno mora imati program. Ako se radi o vrlo jednostavnim uređajima kao što je, na primjer, elektronska vaga, program je jednostavan i obavlja svega nekoliko funkcija kojima se upravlja specijaliziranim tipkama na pradnjoj ploči uređaja.

Složeniji uređaji moraju sadržavati barem potprograme za testiranje i dijagnostiku kvarova uređaja, a često su i programabilni (bilo od strane servisera, bilo od strane korisnika). Programska podrška takvih uređaja u pravilu sadrži jedan zajednički, opći dio kojeg čine moduli za komunikaciju s korisnikom i moduli za povezivanje svih funkcija uređaja.

Takvi zajednički dijelovi se često nazivaju *monitori*.

Isti se naziv često koristio kod prvih mikroračunala koja su se sastojala samo od CPU, RAM-a i male heksadecimalne tastature i display-a. Najčešće su bili dostatni tek za učenje, ali još se i danas mogu naći primijenjeni u nekim industrijskim ili laboratorijskim uređajima, najćešće mjernim.

Monitorski program takvog uređaja morao je omogućiti elementarnu komunikaciju s korisnikom: upisivanje programa/podataka u memoriju, pregledavanje programa/podataka te izvođenje upisanog programa.

Slične su programe imala i mini računala sedamdesetih godina. Obično su se programi unašali bušenom trakom ili karticama, ali po uključenju bilo je potrebno učiniti nekoliko osnovnih koraka te nakon toga također Učitavati i izvoditi programe. Ti su se programi nazivali *monitori*, *kontroleri* i sl.

Pojavom floppy, a kasnije i hard, diskova računalima je bio potreban neki osnovni program kojim će korisnik pogledati sadržaj diska, upisati program ili podatke te izvesti program.

Svako računalo opće namjene ima takav program i zajednički im je naziv "O*peracijski sustav*" (OS).

Što radi i sadrži operacijski sustav

Iz prethodno navedenog moglo bi se zaključiti da je osnovna namjena OS-a komunikacija s korisnikom. To je samo dio istine. OS ima ulogu da odvoji korisnika i *aplikacijske programe* od sklopova računala.

Kod suvremenih računala opće namjene, aplikacijski programi nikad ne pristupaju direktno sklopovima (hardware-u) računala. oni to uvijek čine preko OS-a. Kada je u programu potrebno pročitati sadržaj neke datoteke na disku, poslati tekst na printer ili terminal, saznati koliko je sati i slično, to se uvijek čini tako da se pozove odgovarajući modul u OS-u. Taj modul se brine kako će komunicirati sa sklopovima i kako će obaviti zadani posao.

U tome je i osnovna prednost, jer u slučaju da se sklopovi uređaja promijene, potrebno je modificirati samo OS, dok sve aplikacije ostaju nepromijenjene. S obzirom da se broj aplikacija za neko računalo mjeri stotinama i tisućama jasno je kakav bi posao bio da se u svima mora mijenjati program kada se koristi drugačiji tip diska ili serijske komunikacije.

Taj je koncept i dalje razrađen, pa umjesto da proizvođać računala ili OS-a prepravlja OS za svaki novi sklop koji neki nezavisni proizvođač izbaci na tržište, OS je rastavljen na dva dijela: jezgru (*kernel*) i *driver*-e.

Driver je skupina programskih modula namijenjen za upravljanje jednim sklopom. Točno je definirano kako jezgra OS-a komunicira s driver-ima i po tim uputama svaki proizvođač sklopova može napraviti odgovarajući driver.

Postoje složene primjene kod kojih je potrebno u jednom računalu "istovremeno" izvoditi nekoliko algoritama koji su u međusobnoj interakciji. Sličan problem su i računala koja koristi više korisnika u tzv. time-sharing radu.

Kako jedan CPU može istovremeno izvršavati samo jedan program, ovakav rad je moguća samo tako da se naizmjence izvršavaju naredbe "paralelnih" programa.

Operacioni sustavi takvih računala (*multi-tasking i multi-user*) sadrže mehanizme koji naizmjence izvršavaju naredbe "*paralelnih i istovremenih*" programa (*scheduler*), a po potrebi programe iz RAM-a prebacuju na disk i obrnuto (*swaper ili pager*).

U ovisnosti o algoritmima primijenjenim u tim mehanizmima OS je podesniji za jednu ili drugu vrstu primjene, npr. bankovne transakcije ili mjerenje i upravljanje procesom.

Multi-tasking OS sadrže i mehanizme za međusobnu komunikaciju i sinhronizaciju "istovremenih" programa (procesa).
To su tzv. semafori, poruke, repovi za čekanje i dr.

MS-DOS

MS-DOS (MicroSoft Disk Operating System) je sigurno jedan od najpopularnijih i "običnim" korisnicima računala najpoznatiji OS. Mnogi koncepti preuzeti su od UNIX-a iako neki nisu u potpunosti realizirani.

Kao posebno interesantno svojstvo MS-DOS-a je to da se driver-i pridružuju jezgri naknadno u toku rada OS-a. To znači da nije potrebno pripravljati posebnu verziju OS-a koja sadrži željene driver-e, već korisnik sam određuje koje će drivere koristiti i kada.

S druge strane MS-DOS nema do kraja riješene neke funkcije posebno vezane uz ulazno-izlazne jedinice (prekidni rad, na primjer). Ta činjenica, a i želja proizvođača da izvuku maksimum iz sklopova i OS-a, navodi mnoge proizvođače programskih alata i aplikacija da u svoje programe ugrađuju module koji direktno komuniciraju sa sklopovima (najčešće video-memorija i komunikacijske linije).

Iako se na taj način poboljšavaju performanse koje stoje na raspolaganju korisniku, to je protivno biti OS-a i drivera, a najgore je da to predstavlja velike probleme kod novih generacija strojeva programske podrške.

UNIX

To je multi-user, multi-tasking OS koji u potpunosti odvaja korisničke programe od sklopovlja i strogo poštuje koncept driver-a. Za dosljednu i potpunu implementaciju ovog OS-a na nekom računalu neophodno je da računalo ima ugrađeno sklopovlje za tzv. *Memory management* (MM).

Ti se sklopovi tako programiraju da korisnički program može koristiti samo onaj dio memorije računala koji mu je dodijeljen i ne može ni greškom ni namjerno pristupiti ostalim dijelovima. Na taj se način sklopovski osigurava da jedan proces ne uništo program ili podatke drugog procesa.

Isti se mehanizam koristi da odvoji OS od korisničkih programa. Tako je jedini način da korisnički program pristupi komunikacijskim linijama, magnetskim medijima, memoriji i drugim procesima da zatraži točno određenu uslugu od OS-a. OS će se pobrinuti da taj posao obavi pod strogo kontroliranim uvjetima.

Tako se sistem ne može "srušiti" ili bilo što pokvariti od strane korisnika. To se može desiti jedino ako postoji greška u samom OS-u ili driver-ima. Iz toga proizlazi velika odgovornost u projektiranju i testiranju OS-a i driver-a.

Najnovije verzije UNIX-a (System V release 4) omogućavaju da korisnik sam skroji OS uključujući po volji ne samo driver-e već i jedan od nekoliko tipova scheduler-a (moguće je napisati i

vlastiti). O tome ovisi ponašanje cijelog sustava pa je moguće primjeniti UNIX i u *real-time* aplikacijama.

Jedno od vrlo interesantnih svojstava UNIX-a (djelomično realizirano i u MS-DOS-u) je tzv. redirekcija i cjevovod (*pipe*).
Radi se o konceptu *standardnog ulaza i izlaza*.

Ako korisnik napiše apliakcijski program tako da se svi podaci uzimaju sa standardnog ulaza, a rezultati šalju na standardni izlaz onda je taj program posebno lako koristiti. Naime, ako samo pozovemo program, ulaz će biti s tastature, a izlaz na ekran terminala. Ako podatke spremimo u datoteku, prilikom izvođenja programa možemo narediti da se ulazni podaci uzimaju iz nje. Slično je i s izlazom koji se može preusmjeriti u datoteku. Kako se do svake ulazno-izlazne linije dolazi preko simboličke datoteke tzv. *device*-a, na isti je način moguće podatke uzimati sa serijske linije ili AD konvertera, a izlaz poslati na DA konverter ili printer.

Kako UNIX može izvoditi više programa "istovremeno", moguće je izdati naredbu da s pokrene nekoliko programa, a da se izlaz jednog spoji (pipe-om) na ulaz drugog i tako redom.

Na taj je način moguće spojiti čitav niz programa (koji svaki za sebe obavlja vrlo jednostavnu funkciju) u lanac koji obavlja nešto vrlo složeno. Taj se koncept u UNIX-u naziva *filteri*.

Tako se aplikacijski program ne mijenja, a sposoban je koristiti bilo koju jedinicu i za ulaz i za izlaz.