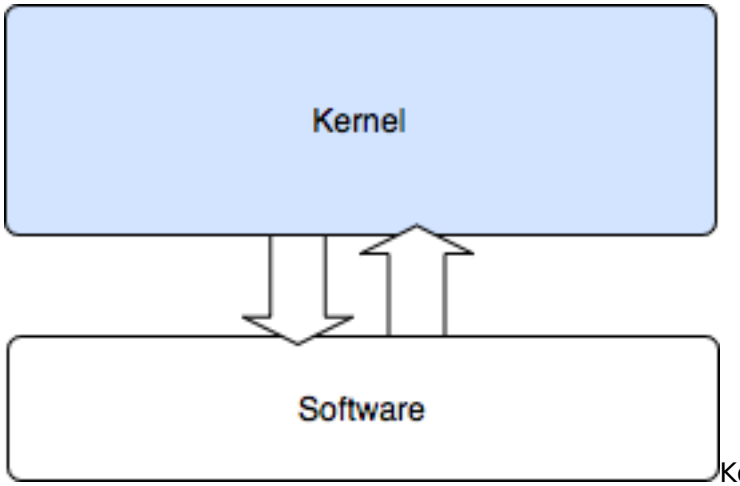
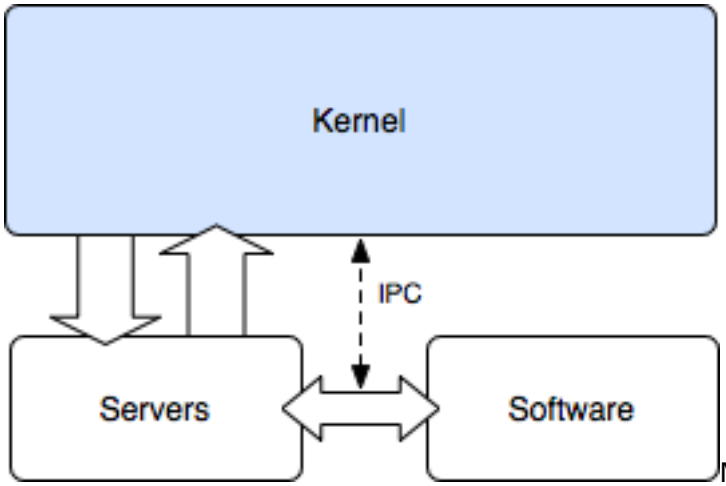
Tipovi jezgre

Što se samog dizajna i naravno same implementacije jezgre tiče, postoji par osnovnih tipova. Dva najinteresantnija i ujedno najviše suprotstavljena tipa su monolitna jezgra (engl. monolithic kernel) i mikrojezgra (engl. microkernel).



Kod monolitne jezgre se cijela jezgra izvršava u jezgrinom prostoru i to upravo u nadglednom načinu rada. Za takvu jezgru je karakteristična vrlo snažna apstrakcija nad sklopovljem kao i sučelje prema aplikacijama sa vrlo bogatim skupom sistemskih poziva i inih primitiva. Unatoč činjenici da su razne funkcionalnosti najčešće implementirane u različitim dijelovima jezgre, stvarno odvajanje takvih dijelova je u praksi nemoguće zbog vrlo visoke razine isprepletenosti samog izvornog koda i podatkovnih struktura. Takva visoka razina međusobne integracije svih dijelova utiče na dobre performanse zbog minimalnih nivoa apstrakcije između samih dijelova kao i visoke efikasnosti koja je posljedica takvog dizajna. Glavna mana ovakvog pristupa je prvenstveno problem da greška u bilo kojem dijelu ovakve jezgre najčešće utiče na pad cijelog sustava, budući da sve komponente nužno vjeruju jedna drugoj. Poznatiji primjeri monolitnih jezgri su: Linux (postojanje modula ne mijenja činjenicu da je riječ o monolitnom dizajnu jezgre), BSD jezgre, Solaris, DOS, većina Unix jezgri kao i Windows jezgra (hibridnog dizajna budući da je dio jezgrinih modula implementiran u vidu korisničkih aplikacija, no smatra se u osnovi monolitnom jezgrom).



Mikrojezgra je sa druge strane jezgra minimalističkog dizajna: aplikacijama se pružaju tek najosnovnije nužne usluge (odnosno sistemski pozivi), a to su već spomenuto upravljanje adresnim prostorom, vremenskim prekidima, dretvama (engl. threads) pojedinog procesa te komunikacijom među procesima. Sve ostale dodatne usluge

poput grafičkih sučelja, mrežnog stoga i sl. su implementirane u korisničkom prostoru u vidu posebnih servisa (engl. servers). Zbog pojave takvih servisa izvan jezgrinog prostora, kod mikrojezgri dodatno dolazi do izražaja potreba za vrlo efikasnom komunikacijom među procesima (engl. inter-process communication) kako je prenošenje poruka jedan od glavnih aspekata rada. Zbog toga što se daleko više poruka prenosi iz jezgrinog prostora u korisnički prostor i obrnuto, mikrojezgre obično imaju nešto lošije performanse od monolitnih jezgri. Spomenuti dodatni servisi su u praksi obične korisničke aplikacije koje se mogu pokretati i gasiti bez loših posljedica za ostatak sustava: jedina njihova privilegiranost je da mogu pisati i čitati neke dijelove memorije koji su nedostupni ostalim procesima. Glavni problem takvog pristupa gdje se neki sistemski servisi mogu gasiti i paliti po potrebi je nestanak strogo definiranog stanja kao kod monolitnih jezgri. Dakle, sasvim je moguće da će u nekom trenutku nestati kakav servis poput mrežnog stoga i da će sve korisničke aplikacije koje su ovisile o mrežnom stogu jednostavno doći u nedefinirano stanje. Iz takvog razloga većina operacijskih sustava koji su bazirani oko mikrojezgri vodi računa o stanju pa omogućava pauziranje aplikacija koje čekaju da se povrati određena usluga i sl. Primjeri mikrojezgri su također prisutni u modernim operacijskim sustavima: Minix, AmigaOS, Mach (GNU Hurd, XNU odnosno Mac OS X), QNX, L4 obitelj, Symbian OS, Singularity.

- Logirajte [1] se za dodavanje komentara

Links
[1] https://sysportal.carnet.hr/sysportallogin