Imamo dva programa, ki tečeta na različnih računalnikih, povezanih med seboj v mreži. Kako lahko program A pošlje programu B polje n celoštevilčnih podatkov? Povejte le, kaj veste o primernem principu take komunikacije.
S pomočjo vtičnic.
Vtičnice so mehanizem za izmenjavo podatkov med procesi. Procesi so lokalni ali na mreži. Ko je povezava preko vtičnic vzpostavljena, se podatki lahko prenašajo v obeh smereh, dokler ena od obeh strani ne zapre povezave.

Koncept:

Strežnik	**Odjemalec**
Vzpostavi vtičnico za poslušanje in čaka na povezavo odjemalca	
	Zgradi vtičnico odjemalca in se poizkuša povezti s strežnikom
Sprejme odjemalčev poizkus povezave	
Pošilja in sprejema podatke	Pošilja in sprejema podatke
Zapre povezavo	Zapre povezavo

Opišite naloge jedra pri operacijskem sistemu UNIX. Kako med seboj komunicirajo uporabnik, procesi, jero in strojna oprema pri tem sistemu?

- Nadzor nad aktivnimi procesi in dodeljevanje spomina
- Procesi dajejo zahtevke jedru (sistemski klici)
- Oskrba procesov s podatki
- Je uporabnik resursov in jih razporeja procesom optimalno (blokiranje, CPU, en proces naenkrat v časovne kvantu 10 ms)
- Zagotavlja značilnost virtualnega stroja (iluzija edinega procesa)
- Je programski vmesnik do sistema (vsebuje gonilnike naprav)
- Sprejema strojne prekinitve periferije
- Delo z virtualnim spominom
- Delo z datotekami
- Protokoli za komunikacijo po mreži
- Delo z več procesorji
- Nitkanje

Uporabnik proces (koda)

Proces jedro (sistemski klici – zahtevki)
Proces jedro (rezultati)

Jedro periferija (zahtevki)
Jedro periferija (prekinitve)

Kaj so naloge lupine in kaj naloge jedra ob izvršitvi ukaza rm moja_datoteka?
Lupina je UNIXov interpreter ukazov.
Lupina razbira in izvaja zaporedja ukazov in programirane akcije (spremenljivke, pogojni skoki, ponavljalne strukture) zapisane v ukaznih datotekah (scripts).

Razložite vlogo sistemskih klicev na sistemu UNIX. Naštejte bistvene razlike pri delu z datotečnim sistemov z uporabo sistemskih klicev in pri delu z datotečnim sistemom z uporabo funkcij knjižnice.
Sistemski klic je zahteva jedru, da nekaj naredi za uporabnikov program.
Le jedro sme imeti direkten dostop do sistemskih podatkovnih struktur, diskov in periferije!

Uporaba sistemskih klicev:

- Delo z datotečnim sistemom
- Nadzor programskih procesov
- Medsebojno komuniciranje med procesi

Delo z datotečnim sistemom pri sistemskih klicih:
Vse nizkonivojske vhodno izhodne operacije so izvršene s sistemskimi klici in sicer z branjem iz datoteke ali pisanjem v datoteko ne glede na vrsto datoteke ali naprave.
Na nivoju jedra so datotekam prirejene številke (file descriptors), ki določajo ustrezne vhodno-izhodne kanale oziroma neformatirane tokove bytov med nekim procesom in datoteko.

Delo z datotečnim sistemom pri funkcij knjižnic (običajno ne rabijo jedra za izvršitev naloge za uporabnikov program):
(prenosljivost, vmesno pomnjenje, prijaznost, konverzija numeričnih podatkov)
V okviru knjižnice so datotekam prirejeni datotečni kazalci, ki kažejo na notranje strukture kjer so shranjeni podatki o datoteki. (datotečni izravnalnik, naslov, dolžina, položaj aktivnega znaka, r/w).

Razložite pozicijske argumente in vidljivost spremenljivk v lupini bash. Kako je s pozicijskimi parametri ukazne datoteke in kako s pozicijskimi pramateri funkcij? Kako je s spremenljivkami ukazne datoteke in kako s spremenljivkami funkcij?
Spremenljivke v skripti so globalne. Spremenljivke, ki so lokalne v funkciji, se deklarirajo s predpono local. Te spremenljivke so vidne samo v okviru funkcije, ki jih vsebuje.

Ko bash skript kliče funkcijo, se argumenti pri klicu funkcije spremenijo v pozicijske parametre. Uporabljamo jih na isti način, kot pozicijske argumente skripte. Argument št.0 (#0 - ime skripte) ostane nespremenjen.

Če se v funkciji izvrši komanda return, se funkcija konča, skripta se nadaljuje takoj za klicem funkcije, pozicijski argumenti pa se vrnejo na prvotno vrednost. Če ukaz return vrne numerično vrednost, je dosegljiv kot izhodni status funkcije, drugače pa je izhodni status funkcije enak statusu zadnjega ukaza pred return.

Kakšne možnosti imamo pri sistemu LINUX, da restavriramo pomotoma izbrisane datoteke?
Če pomotoma pobrišete pomembno datoteko na vašem sistemu, ni nobenega načina, da jo ,,povrnete``. Včasih pa lahko prepišete ustrezne datoteke z diskete na vaš trdi pogon. Če, na primer, zbrišete /bin/login, ki vam omogoča prijavo, preprosto zaženite sistem z zagonske/korenske diskete, priklopite korenski datotečni sistem na /mnt in uporabite ukaz

cp -a /bin/login /mnt/bin/login

Izbira -a naroči cp, da ohrani dovoljenja datotek, ki jih prepisuje.

Seveda, če datoteke, ki ste jih zbrisali, niso potrebne sistemske datoteke, ki imajo ustreznice na zagonski/korenski disketi, potem nimate sreče. Če redno delate varnostne kopije, jih lahko obnovite iz njih.

Razložite mehanizem nastanka in mehanizem končanja procesa, ki teče v ospredju na operacijskem sistemu UNIX. Kateri sistemski klici se uporabijo?
Pri incializaciji sistema jedro generira nekaj začetnih procesov (pid 0 - scheduler; pid 1 - init). Ostali procesi so generirani z dvema osnovnima sistemskima klicama za oblikovanje okolja procesov, fork() in exec().

fork() – naredi dvojnika procesa; stari proces – predhodnik; novi proces – naslednik
exec() – zamenja sliko procesa z drugim procesom
exit – konča z delom (vrne predhodnem procesu status procesa)
wait - sistemski klic omogoča čakanje predhodnika, da njegov naslednik zaključi z delom
kill – ubije proces (sigkill = exit; Sigstop = stop, sigcont

Fora je v temu, da mu jedro nameni več resursu. Namreč procesi, kateri so v ospredju imajo prednost pred tistimi, katerimi so v ozadju. Se pravi proces, ki je v ospredju mu je namenjen več pomnilnika, hitrosti, kot tistemu, kateri je v ozadju. Nekako tako! Ima prednost pred ostalimi, saj je trenutno aktualen.

Kaj je zombi in kaj sirota?
Sirota je proces pri katerem se je starševski proces zaključil ali se prekinil. Sirota se dodeli procesu init. Četudi je potem proces določen staršu je še vedno sirota.
Če naslednik končna z delom prej preden je predhodnik izvršil sistemski klic wait, se deloma razkroji in se tako okrnjen ohrani; oziroma postane zombi.

Aktivni znaki in razlike med narekovaji.
Aktivni znaki:
<, > - preusmeritev toka
| - usmernik
$ - spremenljivka
Narekovaji (levi, desni, dvojni)
\ - prepoved aktivnosti

Narekovaji:
Dvojni – ohrani aktivnost nekaterih znakov (\ , $, narekovaji)
Enojni levi – striktno kot string znakov
Enojni desni – kot ukaz

Zagon OS, kako je to pri linuxu?

1. Inicializacija jedra
    a) Preveri sistemske naprave
    b) Identifikacija specifičnih naprav
    c) Preskus bistvenih naprav
    d) Preskus strojnih podsistemov
    e) Inicializacija datotečnega sistema
    f) Spreminjanje konfiguracije jedra
2. Init()
    a) Init() začne življenje kot nit jedra in končna kot proces na uporabniškem nivoju (/sbin/init)
    b) Je predhodnik vseh procesov
    c) Teče v uporabniškem načinu (do jedra dostopa preko sistemih klicev)
3. Najava uporabnikov (login)
4. Zaustavitev
    a) Zapis »bufferjev«: da ne bi izgubili podatkov in okvarili datotečni sistem
    b) Zahteva od »init«, da pošlje vsem procesom signale SIGTERM in SIGKILL (jih torej konča)
    c) Zaporedje zaustavitve
        - Shutdown
        - Blokirano je ponovno prijavljenje
        - Vsem procesom pošljemo signal SIGTERM in jim tako povemo, da se sistem ugaša
        - Procesi se čisto zaključijo
        - Procesu init pošljemo signal, da naj spremeni nivo izvajanja

Razložite pomen in delovanje nalagalnikov, ki jih srečamo pri zagonu operacijskega sistema (Loader, Bootstrap Loader). Kaj je funkcija LILO oziroma Grub?
Nalagalnik (loader) je program, ki kopira kodo (običajno z diska) v pomnilnik in spoži izvajanje te kode.
Bootstrap Loader je program ki naloži »prvi program«. Pogosto je večstopenjski – primarni, sekundarni.

LILO in GRUB sta sekundarna zagonska nalagalnika. Sekundarni zagonski nalagalnik naloži zagonski sektor particije.

LILO:
- Leži v MBR ali zagonskem sektorju particije
- Ne ve nič o datotečnem sistemu
- Omogoča izbiro jeder Linuxa
- Omogoča zagon jeder, ki niso Linux

GRUB:
- »Lupina« za čas zaganjanja
- Sproti izvaja novo konfiuracijo
- Lahko zagajnamo druge OS

Razložite, kaj pojmujete pod konfiguriranjem operacijskega sistema. Razložite tudi pomen registra (Registry) pri operacijskih sistemih MS Windows. Kako ga spreminjate?

Narišite diagram prehajanja stanj procesa na večopravilnem sistemu, vsako stanje pa na kratko opišite.

V kakšin vlogah je lahko vse win 2008 server, pa kaj je to active directory?

Kaj so spletne storitve, kateri protokoli se uporabljajo, kako so predstavljeni podatki v spletnih storitvah.

Razloži, v čem se razlikuje klic oddaljene procedure (RPC) od klica navadne funkcije.

Primerjajte operacijska sistema WIN NT in UNIX. Kaj imate skupnega? V čem se razlikujeta?

Naštejte vsaj tri zlonamerne kode in opišite razlike med njimi. (Virusi, črvi,...)

divjak:
raid, spooler, buffer overflow, active pages, dns porazdeljeni sistemi, nfs, member member off, buffer owerflov, inštalacija OS-a, vprasanja iz prosojnic! (kljuci, ntfs,stripped value, konfiguracija op.sistema)

jager:
komunikacije med uporabnikom, procesi,datotecni sistem, vsebina tabele-i vozlisca, procesi (fork,zombi,sirota), signali, zacetni procesi, kakšne spremenljivke ma bash (globalne in lokalne), aktivni znaki v bashu,ponavljalne strukture v bashu (for, while, until...predvsem so važne razlike med while in until), ukaz expr...