Lösningsförslag till Tentamen i Operativsystem 2G1520 D, IT, E mfl. och 2G1504 för I Operativsystem
Frågor: (som normalt bara kräver kortfattade svar 1-10 rader)

1) POSIX standarden föreskriver inte att signaler skall köas. Beskriv hur man i en kommando tolk ändå via en egen signalhanterare för SIGCHLD garanterat alltid kan upptäcka alla bakgrundsprocesser som terminerat även om man tidvis blockerar signalen. Beskriv lösningen både i ord och pseudokod. (2p)

Eftersom flera bakgrundsprocesser kan terminera medan en signal är blockerad och signalerna inte garanterat köas så riskerar man att signalhanteraren bara att startas en gång även om flera processer terminerat. Däremot finns informationen om vilka barnprocesser som förändrat status (exempelvis terminerat) tillgänglig för kommandotolken och kan läsas av med wait()/waitpid(). Lösningen är därför att i signalhanteraren polla, icke-blockerande, med waitpid() och kontrollera status för de processer som föraändrat status för att se om någon/gra av dessa terminerat.

while((pid=waitpid(-1, &status, WNOHANG) > 0) if(WIFEXITED(status)) PRESENTINFO(pid, status);

2) Vilka är de grundläggande skälen till att man infört/har processer i praktiskt taget alla moderna operativsystem? (1.5p)

Förbättrad prgrammeringsmodell
Felisolering till processerna
Möjlighet att tillåta flera samtidiga användare
Möjlighet till bättre resurtsutnyttjande genom överlappning av I/O och beräkningar

3) Relokerbar kod, länkare och laddare är begrepp de flesta känner till. Men under vilken epok i datorhistorien introducerades de och vad var skälet till att de behövdes? (1p)

DE här begreppen introducerades redan i den först agenerationen datorer då man insåg att det var praktiskt att inte behöva skriva om samma kod flera gånger utan att det var bättre att lägga rutiner i bibliotek. För att sedan kunna använda dessa rutiner i andra program krävs både en länkare4 och en laddare

4) Antag en fysisk adressrymd om 2^{20} och att man har virtuellminne med sidhantering där man använder en sidstorlek på 1K. Till vilken fysisk adress översätts följande (binära) virtuella adress: 00000001010111100111 givet följande sidtabell: (4,12,3,5,7,1,2,6,5,8) Indexering sker som i C. Svara Binärt! (1p)

10 bitar används för att indexera i sidan övriga 10 bitar för att indexera i sidtabellen. De 10 bitar i adressen som används för att indexera i sidtabellen representerr talet 5 (0000000101). Index 5 i sidtabellen talar om att sidan ligger i ram 1, dvs den fysiska adressen blir 00000000010111100111

5) Givet ett primärminne med 4 ramar hur många sidfel får man för följande refrenssträng vid ren demand paging för sidutbytesalgoritmerna FIFO, LRU och OPT? Referenssträng: 1,3,6,8,2,4,1,7,9,4,6,2,6 (1.5p)

FIFO 11 sidfel
```
XXXXXXXXX-XX-
1368241799622
 136824177966
  13682411799
   1368244177
```

LRU 11 sidfel
```
XXXXXXXXX-XX-
1368241794626
 136824179462
  13682417944
   1368241799
```

OPT 8 sidfel
```
XXXXXX-XX----
1111114446266
 366646662622
  38262224444
   3821799999
```

6) När kan man anta att rättigheter i ett filsystem, som läs/skrivrättigheter, kontrolleras om man har ett distribuerat filsystem med en tillståndslös server respektive en server med tillstånd. Har dettta någon betydelse för säkerheten i systemen? (1p)

När man kommunicerar med en tillståndslös filserver så kommer man att behöva skicka all information för varje begäran: om användare, vilken fil som skall accessas och hur, eftersom filen måste öppnas varje gång. Då kan man anta att accesskontroller utförs i varje begäran.

För en filserver med tillstånd hålls normalt filen öppen till dess användaren stänger den. Då kontrolleras bara accessrättigheterna en gång per session som filen används i (filen kan iofs öppnas och stängas av användaren flera gånger).

Det senare fallet kan tillåta att en användare kan accessa en fil som han/hon inte längre har accessrättigheter till, om accessrättigheterna modifierats efter det att användaren öppnade filen.

7) Rangordna via vilken mekanism man snabbast kan accessa sista datablocket i en stor fil om man använder: FAT, i-noder, allokering av diskblock med länkad lista. Förklara ditt svar! (1.5p)

Snabbast först: FAT man kan accessa diskblocket via en (1) indexering i FAT, ii) i-noder garanterar att man maximalt behöver göra ett litet begränsat antal indexeringar i den trädliknande struktur som i-noderna använder för att hitta blocket, iii) indexerad allokering, med länkarna i värsta fall i diskblocken, kräver N-1 indexeringar.

8) Var bör information om accessrättigheter för filer och directoryn lagras? i) i roten på filsystemet, ii) i varje directory eller iii) tillsammans med varje fil? (2p)

För att underlätta att hålla accessrättigheterna konsistenta bör dessa lagras tillsammans med filen respektive directoryt. Lägger man accessrättigheter för en fil i ett directory kan man i värsta fall få situationer där accessrättigheterna ser olika ut beroende på via vilket directory man accessar filen.

9) Vilken information måste finnas i monteringspunkten för ett monterat filsystem? (1p)

Att det är ett monterat filsystem, var filsystemet finns (på en lokal disk eller hos en server), typ av filsystem och hur det accessas

10) Förklara för var och en av följande: användarkommandot `rm`, biblioteksfunktionen `unlink()` och systemanropet `unlink()`, om/varför det kan finnas en eller flera instanser/implementationer av dem i ett system och hur man i så fall vet vilken version som används! (3p)

Systemanrop finns bara i en instans/implementation eftersom de är en del av kärnan. Både användarkommandon och biblioteksfunktioner kan det finnas flera instanser/implementationer av, jfr att man t.ex kan ha olika kompilatorer med olika bibliotek som SUN, GNU mfl på en och samma maskin.

11) I många hårddiskenheter finns en buffert/cache. Beskriv två vanliga användningar för den och hur de påverkar prestandan! (2p)

Två vanliga användningar är: i) att köa läs/skrivbegäran och ev ordna dessa så att disken utnyttjas effektivt; ii) för att göra read-ahead vid läsningar och i bufferten/cachen lagra data för snabbare åtkomst (läsning)

12) a) Beskriv hur man kan implementera sk. "Stable storage" med hjälp av två separata hårddiskar! (1p)

Vid skrivning skriv till bägge diskarna (en i taget) till dess att man kan verifiera att bägge skrivningarna lyckades eller om man inte lyckas inom ett begränsat antal försök så signalera fel.

Vid läsning läs från en av diskarna och kontrollera checksumman om den är fel prova att läsa och kontrollera cehksumman från den andra disken. Om bägge checksummorna är fel så signalera fel.

b) Hur stor blir sannolikheten att en skrivning går förlorad om man har två obeorende diskar där sannolikheten att tappa en enskild skrivning till en disk är 10^{-9}? (0.5p)

med oberoende antagandet 10^{-18}

13) Jämför trådar i användarläge, trådar i kärnläge respektive processer med avseende på: i) möjlighet att bygga feltoleranta system, ii) portabilitet (2p)

Feltolerans: Användartrådar - OS:et känner inte till trådarna och ett fel i en tråd påverkar hela processen
Kärntrådar: Os:et känner till trådarna och det finns möjligheter att låta ett fel bara påverka en tråd men i många fall påverkas hela processen
Processer: Är den naturliga del i vilken fel normalt sett kapslas in. dvs ett fel i en process påverkar normalt inte andra processer. Lämpar sig väl för att bygga feltoleranta system.

Portabilitet: Så länge systemen är baserade på standarder som t.ex POSIX är portabiliteten oftast inget större problem.

14) I distribuerade system väljer man oftast att inte adressera en server/klient med dess PID utan man använder portnummer. Varför? (1p)

PID kan oftast inte bestämmas av användaren när en process skapas - därför är det svårt för en icke-lokal användare att veta vilken PID en server/klient har, dessutom kan dessa bytas om man tvingas starta om procesen. Portar är däremot en ändpunkt för kommunikation och portnummer kan reserveras och knytas till en process. På så sätt kan man skapa välkända (standardiserade) portnummer som används för en viss tjänst i t.ex hela Internet.

15) Vid komunikation med meddelandesändning kan send-primitivet vara antingen icke-blockerande eller blockerande. I praktiken visar det sig ofta att icke-blockerande send-primitiver kan blockera under kortare perioder. Förklara vad som kan ligga bakom detta beteende! (1p)

Icke-blockerande send skriver normalt till en buffert. Från bufferten läses sedan och skickas meddelandena til mottagaren. Detta kan ske t.-ex över ett nätverk och meddelandet kan då inte skickas förrän nätverksanslutningen hinner skicka meddelandet. Det är för att dölja sådan väntab som bufferten finns.

I ett sant icke-blockerande send så kan man skriva över innehållet i bufferten om t.ex nätverket inte hinner med. I många fall har man därför en synkronisering på bufferten så att den inte kan skrivas över - därför kan man ibland uppleva att ett icke-blockerande send faktiskt blockerar i väntan på att det skal bi ledig plats i bufferte. Jfr synkroniseringen i UNIX pipes

16) a) Beskriv "dining philosophers" problemet! (0.5p)

Ett runt bord med en skål spaghetti i mitten. N sitplatser med en gaffel vid varje plats. En filosof som vil äta sätter sig på en ledig plats och behöver två gafflar för att kunna äta. Filosofen tar upp de två närmaste gafflarna. Får filosofen inte tag i en gaffel väntar denne till dess gaffeln blir ledig.

b) Ge två olika lösningar till problemet och bevisa att de undviker baklås (deadlock) (2p)

Man låter maximalt N-1 filosofer äta samtidigt. Då undviks cirkulär väntan då filosofer på var sida om den lediga platsen inte kommer att vänta på varandra, dvs minst en filosof kan alltid ta två gafflar och på sikt bli klar.

Man numrerar gafflarna och låter filosoferna ta gafflarna i strikt nummerordning - då bryts också villkoret om cirkulär väntan.

17) a) Förklara begreppet live-lock! (1p)

Live-lock inebär att två eller flera processer synkroniserar med varandra på ett sådant sätt att de inte kommer framåt i sin exekvering. Dvs de exekverar men gör inget nyttigt arbete.

b) Om Kalle och Stina vill ringa till varandra finns då någon risk för att de ska råka hamna i live-lock? Vad gör att risken finns/inte finns? (1p)

Ja om bägge slår tefonnumren ungefär samtidigt kommer det att tuta upptaget för bägge eftersom den andres linje är upptagen. Lägger då bägge på och försöker ringa om igen samtidigt kan man hamna i ett live-lock

18) Du vill kontrollera om ett program, som du inte har tillgång till källkoden till, kan vara en trojan eller innehålla ett virus och vad det i så fall gör. Vilka standardverktyg/systemprogramvara kan då vara lämpliga att använda i ett UNIX/LINUX system? Och hur kan de vara användbara? (det handlar alltså inte om anti-virus programvara eller barndväggar och vi antar att du kör det på en maskin utan kontakt med omvärlden där ev bieffekter av viruset/trojanen kan försummas). (2p)

Verktyg som TRUSS/STRACE tillåter att man kan se vilka systemanrop ett program gör och parametrar till dessa - ovärderligt eftersom ett program inte kan göra mycket skada annat än via systemanrop. En disassembler kan också vara bra för att återskapa assmblerkoden så att man mer i detalj kan se hur programmet beter sig. Har man inte tillgång till dessa verktyg kan man i vissa fall få ut en del intresant information genom att köra programmet i en debugger..

19) Vilka av följande filtyper kan dölja/innehålla ett virus?
    i) ASCII-text
    ii) postscript
    iii) MS-Word (1.5p)

Samtliga: ASCII kan inehålla källkodsvirus som aktiveras då källkoden kompileras, länkas och körs. Postscript är ett programmeringsspråk som kan interpreteras/exekveras och ett MW-Word dokument kan innehålla kod som körs i form av macron.

20) * Vad är det som gör att man i många sammanhang ofta föredrar Rate Monotonic Scheduling före Shortest Job First som schemaläggningsstrategi för realtidsystem trots att SJF kan schemalägga större laster? (1p)

Shortest Job First är inte en realtidsschemaläggningsmetod då den inte tar hänsyn till deadlines (som t.ex EDF) därför är RMS att föredra. SJF riskerar till exempelk att svälta ut stora processer, dvs. göra så att de svälts ut.

21) a) Var implementeras användargrafiken (t.ex fönsterhantering) i Microsofts OS som t.ex W2K respektive i LINUX? (1p)

MS: I OS kärnan
LINUX: i klienter/servrar på användarnivå

b) Vad har strategin/strategierna för för- och nackdelar? (1p)

Att implementera grafiken i kärnan kan möjligen göra den effektivare - men samtidigt mindre flexibel då alternativet att göra grafiken som klienter/servrar på användarnivå tillåter att man enkelt byter ut grafikpaketen vid behov.

"Det krävs ett helt nytt sätt atttänka för att lösa de problem vi skapat med det gamla sättet att tänka"
Albert Einstein