Handbok för SvenskMUDmagiker

ett hjälpmedel för byggarna i SvenskMUD

Linus Tolke, Gud

För pappersutgåvan gäller dessutom följande speciella förbehåll: Om du tycker att du har någon nytta av den så sätt in 30 kr på postgirokonto 483 11 40-1. De pengarna går oavkortat till Linus skaparen (gud) som ur egen ficka i förväg betalat tryckkostnaden. Ge annars bort ditt exemplar till en annan SvenskMUDmagiker.

Om du kopierar pappersutgåvan så ta inte med detta speciella förbehåll.

Första utgåvan. Revision 2.

Förord till första utgåvan:

Målet med denna bok är att vara en första hjälp till nya magiker i SvenskMUD att komma igång med programmeringen. Betoningen är på vad man bygger för saker och hur objekt samarbetar i spelet och boken innehåller inte något alls om hur LPC-språket fungerar och används. Det gör boken väldigt SvenksMUD-specifik.

SvenskMUD har under våren 1993 varit utsatt för stora förändringar. Det är egentligen ingenting som berör det som står i denna bok utan det har mer varit en acceleration i byggandet. Det är en av orsakerna till att jag tyckte det var viktigt att skriva denna bok. En annan viktig orsak är att jag tycker inte att anslagstavlorna i spelet, brev-skickande och pratande fungerar tillräckligt bra som utbildning för nya magiker. Jag vill göra någonting mer och denna bok är det bästa jag kunde komma på.

Sedan revision 0 har det tillkommit en del kapitel men framför allt är hela boken numera ett texinfo-dokument istället för framemaker och det innebär att du kan hämta den infoiserade versionen och ha den i din emacs.

Sedan revision 1 har det nya kapitel 1 tillkommit med fokusering på SvenskMUDs plats i världshistorien. En del andra kapitel eller stycken har bytt plats.

Vill du ha fler exemplar, eller SvenskMUD-knappen så hör av dig till författaren Linus.

Tack till:

Datorföreningen Lysator
Alla magiker i SvenskMUD

1 SvenskMUDs plats i litteraturen

Under 1980-talet utvecklades ett internationellt nätverk av datorer i högskolor och universitet och även vissa företag. Det består av snabba unixdatorer och en fundamental byggsten är snabb och direkt överföring av information mellan nästan godtyckligt valda datorer i nätverket.

Ett sådant här nätverk är en perfekt förutsättning för göra datorspel med en fleranvändardimension. I den ursprungliga nätverket är kapaciteten ganska liten så de första spel som uppenbarar sig är textbaserade[1]. De ser mycket ut som de adventurespel som förekom under 1970-talet och alla kommandon och beskrivningar ges med enkel text.

Den här typen av spel kallas mud.

Man kan hävda att mud bara är spel men man kan också hävda att de är en ny sorts litteratur där läsarens (eller spelarens) och andra läsares val i varje läge påverkar hur historien kommer att fortsätta.

Det finns likheter med traditionell litteratur vad det gäller den skönhet och det intellektuella utbyte som man kan få ur spelen och också i hur muddet speglar författarens upplevelser om samtiden.

Till skillnaderna hör det sätt på vilket intrigen byggs upp. I traditionell litteratur är läsaren (deltagaren) utlämnad helt åt författarens val och författaren har full kontroll över hur handlingen byggs upp och förs framåt. I ett mud är det läsaren (deltagaren) och hans eller hennes med-läsare (andra spelare) som driver fram handlingen och bestämmer hur de olika intrigerna vävs samman och påverkar varandra.

Att vara författare till en sådan här flervalshistoria är mycket svårare än att författa en vanlig bok, dvs om den skall uppnå samma kvalitet. Till dags dato har inget spel ens lyckats närma

[1] Det finns nu i början av 1990-talet redan flera olika diskussioner igång om hur man bäst gör motsvarande spel med grafik och sk. virtual reality och vi har redan sett ett par exempel på detta: BSX-mud, Doom

sig ”kvaliteten” som finns i den enklaste kiosklitteratur. Mest för att det är svårt att åstadkomma en intressant intrig.

Å andra sidan är det många saker som finns i mud som aldrig kan realiseras i en vanlig bok. Bl.a. kan författaren själv finna nöje i att ”läsa spelet” eftersom intrigen kan förändras av andra ”läsare”.

SvenskMUD hör till en typ av spel som kallas LPmud. Det är en typ som utvecklats av datorintresserade och en av grundideerna är att det skall vara lätt att skapa mer saker i spelet (skriva mer i boken). Detta gör att spelet förändras från dag till dag. Det tillkommer saker och saker ändras.

Många mud fungerar så att det finns möjlighet för läsarna (spelarna) att efter ”slutläst” mud, oftast en viss mängd avklarade uppdrag i spelet, få möjlighet att själva ta del i skapandet. Detta skänker ytterligare en intressant dimension av gemensamt författarskap[2] som är väldigt sällsynt i den traditionella litteraturen.

Egentligen är skillnaden mellan att vara ”läsare” och ”författare” lite diffus eftersom ”läsaren” också påverkar historien. I SvenskMUD skiljer man på dödliga och magiker. Magiker har rätt att skapa helt nya saker medan de dödliga är utlämnade åt de möjligheter som magikerna skapar åt dem.

Ibland kallas världen som finns i spelet för en virtuell värld och de olika magikerna ansvarar kanske för var sitt geografiskt område i den virtuella världen.

1.1 Målet med SvenskMUD

Det ursprungliga målet med SvenskMUD är att erbjuda ett alternativ till den engelskspråkiga amerikainfluerade kulturen som finns i den svenska hackervärlden och speciellt bland utbudet av muddar. SvenskMUD var faktiskt först i världen med att erbjuda ett mud på Internet där själva kommunikationsspråket var något

[2] Det rör sig i SvenskMUDs fall om ca 20 samtidiga aktiva skapare

annat än engelska. Sedan dess har det dykt upp mud på tyska också.

Världen är tänkt att återspegla den nordiska mytologin och den svenska litteraturen och utspela sig någongång på 1800-talet.

1.2 Problem med detta mål

Med ständigt ca 20 aktiva författare och en ganska jämn ström av nytillkomna sådana är det väldigt svårt att hålla en värld som är konsistent vad det gäller tidsepok och geografisk utbredning i den virtuella världen. I synnerhet som en del nytillkomna magiker gärna vill skapa saker som direkt märks för spelarna och placerar sina första saker så att spelarna formligen snubblar över dem[3].

Dessutom har en del magiker svårt att skaka av sig oket från den amerkanska kulturen och det finns inslag som inte riktigt lever upp till kraven/målen.

Man måste vara medveten om vad det är för personer som är spelare och blir magiker. Eftersom spelet bara är tillgängligt för personer som finns på Internet så är det nästan uteslutande datorintresserade högskolestuderande som spelar. Det innebär att mycket av det skapande som görs inte handlar om att få en bättre värld utan istället är inriktat på att hitta programmeringtekniskt utmanande problem och lösningar.

1.3 Stilen

Ett annat mål är att behålla stilen genom hela muddet. Den definierade stilen är att det skall vara någonstans kring sent svenskt 1800-tal med lite fantasy-inslag. Det innebär att avancerade vapen, vikingar, familjen Hedenhös mm inte riktigt passar. Vill man förlägga sitt område till en annan tidsperiod kan man ju se till att konstruera en portal mellan tidsperioderna och på så sätt fixa en ”förklarlig” övergång till en annan värld.

---

[3] Ännu mer irriterande är det eftersom de första sakerna en ny magiker gör sällan håller hög kvalité.

1.4 Organisatoriska problem

Det vore ganska enkelt att höja kvaliten på spelet genom att införa högre krav på magikernas kunskap, kräva att varenda pryl som magikerna skapas skall godkännas eller hänsynslöst rensa ut prylar som inte passar och magiker som inte håller stilen.

Sådana åtgärder orsakar genast motsättningar i författargruppen och resulterar otvivelaktigt i att det fungerande resultatet blir mycket mindre. Mindre diversitet och mindre möjligheter. Dessutom finns det ingen som har tid att ställa upp med den tid som behövs för det trista arbetet att godkänna och rensa ut prylar.

I SvenskMUD är målet att skapa en kreativ och hjälpsam anda bland magikerna. I viss mån har stilmålet och kvalitén fått kompromissas för att alla skall känna att de är välkomna att deltaga i skapandeprocessen.

Geografiska inkonsistenser i den virtuella världen löses genom att övertala magikern att flytta sitt område istället för att flytta det utan förvarning. Inkonsistenser i tidsepoken löses genom att man skapar en tidsmaskin eller genom att man låter spelaren börja drömma och i drömmen vara i en annan värld eller liknande.

Allt för att så många som möjligt skall få utlopp för sin kreativitet och känna på denna alldeles speciella typ av författarskap.

Ett angränsande problem är att författarna är spridda över hela Sverige (egentligen hela världen) och i stor utsträckning inte känner varandra. Att kommunikationen dem emellan är begränsad till att prata med varandra i spelet i den mån de råkar vara inne i spelet samtidigt och skicka brev till varandra gör det ibland kan vara svårt att förstå varandra. Det blir lätt missförstånd.

1.5 Gud påverkar

Pga ovanstående kommunikationsproblem kan det inte skapas en fungerande demokrati i spelet utan jag har utnyttjat min position som gud och implementerat mina ideer. På senare tid har jag mest utnyttjat min diktatorställning till att utse Ärkemagiker som jag tror kan behålla den kreativa miljön och arbeta för konsistens i spelet. I detta val har jag också försökt verka för att SvenskMUDs underhållningsvärde skall öka och skapandet skall koncentreras på en fin värld.

De ideer som jag implementerade ursprungligen är:

- Alla magiker får läsa överallt, det är alltså lätt att se hur andra gör sina objekt och saker. Jag tror att detta är ett väldigt bra sätt att lära sig hur man gör olika saker.
- Så stor frihet som möjligt för att inte sätta käppar i hjulet på kreativiteten. I vissa avseenden är det i SvenskMUD väldigt högt i tak vad det gäller vad en magiker får göra och inte får göra.
- Alla magiker kan göra lika mycket/lite i spelet. Magiker kan spionera på dödliga men inte på andra magiker. Det finns i spelet få tekniska begränsningar som skiljer ut magiker på olika nivåer. Det är svårt att avgöra hur detta uppfattats bland magikerna. De flesta verkar inte förstå att jag och ärkemagikerna inte tvunget är bättre programmerare utan förutsätter att vi besitter överlägsen intelligens och kunskap.
- Målet med muddet är att erbjuda intressant underhållning för de som spelar. Underhållningsvärdet ökar om spelet hela tiden lyckas stimulera spelarens uppfinningsrikedom och nyfikenhet. Siffror som rapporterar tekniska detaljer om spelet, tekniska kommentarer mm. stör skönheten i mudvärlden. Eftersom spelarens egenskaper är implementerade som siffror och kroppens funktioner bara är matematiska operationer på dessa siffror är det lockande för den late författaren att bara rapportera dessa siffror rakt av till spelaren. Jag vill att underhållningsvärdet i SvenskMUD skall finnas i de saker som skapas och i historierna som utspelas och att det inte som i andra mud ska bli en jakt på siffror och därför har jag i flera olika steg försökt få bort siffrorna ur spelarnas värld.

2 Om SvenskMUD

2.1 Drivern och var kommer den ifrån

SvenskMUD är ett LPmud. Det innebär att drivern är av samma typ som den driver som Lars Pensjö vid Chalmers dataförening CD först skrev. Den interpreterar ett C-liknande språk kallat LPC. SvenskMUD kör för närvarande en lite justerad version 3.1.2. av drivern.

Den första LPmudden kom i april 1989 och det var Genesis som kördes på milou vid CD. Det var skrivet i LPC och spelets språk var engelska. I maj 1989 så hade även föreningen Lysator ett mud, då körandes på brutalix. Det är detta mud som sedermera blev det legendariska nannyMUD, som är det för närvarande äldsta muddet som bygger på det ursprungliga mudlibbet och har magiker kvar från allra första början.

2.2 SvenskMUDs historia

SvenskMUD såg dagens ljus den 29 juli 1991. Det har utvecklats genom översättning av nannyMUDs mudlib. Precis i början fanns det till och med vissa objekt och rum som var de urpsrungliga engelskspråkiga objekten vilket ledde till en hel del lustiga texter.

Det som gjordes var en direkt översättning vad gäller viktiga rum och funktioner, kyrkan, puben, spelarobjektet mm, men jag passade på att bygga om byn så att den fick en lite annorlunda utformning, dvs rummen sitter ihop på ett lite annorlunda sätt. Exempel på detta är att man går ur kyrkan västerut, man kommer till Leo från äventyrarnas klubb istället för från kyrkan.

När vi väl hade gjort det mesta av arbetet med översättningen så var vi tvungna att lägga till en del saker i mudlibbet för att det skulle passa bättre med det svenska språket. Bl.a. så har `set_gender` fler alternativ. Ett annat exempel är hantering av namn i bestämd form. `query_namnet` och att funktionen `set_name` tar två argument eller en array.

2.2.1 Ny driver

I januari 1992 så gjordes ett byte från COMPAT[1] till NATIVE[2]. Det innebar att en hel del objekt måste skrivas om och det är bl.a. nu som master-objektet kommer till användning. (I kompatibilitetsmod finns inget masterobjekt.)

2.2.2 Lite småpatchar i drivern gör SvenskMUDdrivern väldigt unik

Under sommaren 1992 så lades det in stöd för iso-8859-1 tecken i drivern. Det innebar att de som har ett kommunikationsprogram som klarar iso-8859-1 tecken samt ett terminalprogram som klarar det kan få tecknen åäöÅÄÖ. För de som kör med äldre utrustning konverteras all utmatning till }{|][\ eller iso-646.

2.2.3 Spelarna strömmar äntligen till

Fram till och med sommaren 1992 hade SvenskMUD levt en ganska undanskymd tillvaro och Harry var under stora tider helt ensam inne i spelet. Under hösten 1992 kom det en stor anstormning spelare från Halmstad, Umeå med flera ställen och de blev också snabbt magiker.

Våren 1993 såg en ny anstormning, nu med folk från Stockholms universitet och de drog även med flera av nästa årskurs som fyllt spelet under hösten. Dessutom tillkom det en hel del spelare från Luleå under hösten.

---

[1] COMPAT eller kompatibilitetsmod. När den nya drivern version 3 kom så kunde man köra den i s.k. COMPAT mode. Det innebär att man kunde använda sina gamla mudlib utan problem.

[2] NATIVE eller ordinarie mod. De nya mudlibben kräver denna mod. I NATIVE finns ett masterobjekt.

2.3 Maskinen som SvenskMUD kör på

SvenskMUD kör för tillfället på '`bodil.lysator.liu.se`' hos Dataföreningen Lysator vid Tekniska Högskolan i Linköping - LiTH.

Bodil är en Sun 4/280 med 32MB ram och SunOS 5.3 som är donerad av Sun.

2.4 Föreningen Lysator

Datorföreningen Lysator vid Linköpings Tekniska Högskola är en studentförening oavhängig Universitetets och Tekniska Högskolans officiella organisation och därjämte varje annan yttre intressesfär.

Lysator är en medlemsorganisation i Förbundet unga forskare.

Lysator har till uppgift att bibringa sina medlemmar mål och möjlighet att fördjupa sig i datorteknik och datorvetenskap och att sprida kunskap om modern datorteknik.

3 Att tänka på när du kodar

Resten av denna handbok är tänkt att försöka ge tips till nya och gamla magiker vad det gäller hur man skapar saker i spelet. Tipsen gäller både tekniska frågor och stilfrågor.

3.1 De tio budorden

En hel del tips kan sammanfattas i dessa tio bud.

Du skall inga andra gudar hava jämte emacs.

Vad är det?

Det står dig nästan helt fritt att välja vilka metoder du vill använda för att editera filer om de bara på något sätt kan samarbeta med de grundläggande metoderna för att få in filer i spelet (`ed` och `ftp`).

Emacs med ange-ftp fungerar bra om bara någon lägger in programmet `ls` hos dig i ditt '`bin`'-bibliotek. Det finns alltså ingen som helst anledning att tillbe avgudar som `ed`, `vi` och `framemaker`.

Du skall använda herren din guds namn.

Vad är det?

Glöm inte att fylla alla dina rum med bilder, statyer och reliefer av Gud.

Dessutom kan du fylla dem med människor, djur och saker. Det ger atmosfär och gör det möjligt för nybörjare att samla ihop lite saker och om du gör monstren pratsamma så kan de ge tips till nybörjaren som stannar och lyssnar.

Inte heller vad det gäller namn skall du hysa några som helst betänkligheter. Se bara till att svartlista de namn du använder och inte använda redan svartlistade namn eller spelares namn.

Tänk på vilodagen så du fixar buggar då.

Vad är det?

Söndagar brukar det inte vara så mycket folk inne i spelet så det är en synnerligen lämplig dag att leta buggar på.

Hedra inte din fader och din moder, för att det må gå dig väl och du må bli odödlig i SvenskMUD.

Vad är det?

Din fader och moder vet inte hur många timmar du sitter och kodar ett visst rum eller område. Behöver de veta?

Du skall dräpa.

Vad är det?

Prova att slå ihjäl alla dina monster som du skapar och verifiera att de kan dö, att de kommer tillbaka vid nästa `reset` genom att anropa `reset` i rummet och kolla så att vikten på liket är rimlig.

Du skall begå äktenskapsbrott.

Vad är det?

Det finns faktiskt ingen som har kodat en möjlighet att gifta sig än i SvenskMUD trots prat om sådant. Tills dess kan du ju försöka lyda detta bud. Att vara otrogen mot sin trolovade räknas inte som äktenskapsbrott i detta fall.

Du skall stjäla.

Vad är det?

Om du inte vet hur du skall göra vissa saker så fundera ut var du sett något liknande och titta hur det är gjort där. Det bästa sättet att lära sig är att kolla hur andra gör saker.

Om du lånar till alla dina saker så glöm för den skull inte att lära dig under tiden.

Du skall bära falskt vittnesbörd mot din nästa.

Vad är det?

Hela världen är ju egentligen en schimär. Se till att hålla skenet uppe för spelarna så länge så möjligt. Undvik att skicka ut text av teknisk karaktär till spelarna. Gör fullständiga meningar som betyder något, tillräckligt fullständiga för att spelarna ska förstå vad det handlar om.

Var speciellt försiktig om du håller på att eka eller ropa saker. Tänk på att spelarna lever i en annan värld och anpassa dina meddelanden till det. Ropa t.ex. ”Nu skall jag ge mig iväg på en lång upptäcksresa i öst och nord.” istället för ”Nu går jag hem och lägger mig.”.

Du skall hava begärelse till din nästas hus och hans rum.

Vad är det?

Spring omkring i de andras områden, kolla vad de bygger, skicka en massa buggrapporter och gör sedan bättre själv.

Du skall hava begärelse till din nästas hustru, vapen och rustningar, även hans oxar och åsnor och allt som är din nästas.

Vad är det?

Även när det gäller de andra magikernas prylar skall du se till att använda dem. Om du vill ha ett likadant monster som en annan magiker gjort är det bara att klona ett sådant och stoppa in det i ditt rum. Förbehållet givetvis att det kan vara bra om den andre magikern känner till det så att han inte, dig ovetande, ändrar filnamn eller andra egenskaper hos det monstret.

3.2 Livet som magiker

Livet som magiker skiljer sig i väldigt stor grad från livet som spelare. Som spelare kan man inte se de tekniska svårigheter och möjligheter som magikerna har. Att göra bra och genomtänkta saker är svårt och kräver mycket tålamod och tid.

Det händer att nyblivna magiker har väldigt stora planer och färdigritade kartor med hundratals rum som de vill göra, men så finner de att de egentligen inte har tid att göra det och lägger ner projektet. Man kanske skall passa på att påpeka att fler rum oftast inte är bättre. Det går att göra mycket på bara ett fåtal rum och det är betydligt bättre att göra ett bra rum där det kan hända saker än 100 tomma rum.

Ur spelets och spelarnas synvinkel spelar det kanske inte så stor roll om dina högtflygande planer inte blir implementerade för att det visade sig att du inte har tid och det är heller ingen som tar illa upp om du ångrar dig och gör något annat istället. För din egen skull kanske det är bäst att ta ett steg i taget och börjar med att göra roliga småsaker medan du lär dig det man behöver kunna för att kunna göra bra saker. Först när du kännt på vad som är möjligt och hur lång tid saker tar är det dags att börja planera större projekt.

3.3 Planer att förkasta

Som skapare har man ett ansvar för att se till att världen blir så bra som möjligt. Det är någonting man slipper som spelare. Det kan vara ganska svårt för vissa begrepp blir helt ställda på huvudet.

Från att pengar på banken har betytt trygghet så är nu problemet hur mycket man skall ta betalt för olika tjänster som man implementerar.

Andra saker att tänka på är att när man som spelare springer omkring så är man inne i historien och då gäller det att få så bra prylar som möjligt. Det är lätt att tänka att när jag blir magiker skall jag minsann göra bättre än det här. Bättre i bemärkelsen kraftigare. Det är ännu lättare att faktiskt implementera en sådan pryl men det riskerar att sabotera balansen i spelet. Se stycke 7.8 [Inflation], sidan 33.

4 Filträdet

Detta kapitel beskriver hur muddets filer är organiserade i ett träd. Normalt kan man som magiker orientera sig runt i filträdet med kommandon som liknar de som finns i unixshellar. `cd`, `ls`, `mkdir`, `rmdir` finns till exempel.

Filträdet är direkt uppbyggt på UNIXens filträd. Drivern gör inte `chroot(2)` men skrivning och läsning av filer som inte ligger under detta träd är trots det helt omöjlig inifrån drivern eftersom alla filnamn kontrolleras.

Detta gör att man kan ha symboliska länkar som pekar ut ur detta filträd.

4.1 Kataloger på toppnivån

Toppnivån kan man se genom att göra `ls /`. Vissa av dessa filer är kataloger och vissa är filer. Om man gör `ls -F /` istället kommer katalogerna att listas med ett `/` efteråt.

4.1.1 '`bin`'

Jag har börjat lite smått med att gruppera kommandon under '`/bin`'. Det hör samman med '`/include/commands.h`' och det ärvs också av spelarobjektet '`/obj/player.c`'.

Iden med att placera varje kommandon i en fil kommer ursprungligen från TMI[1] där den är helt genomförd.

Vinsterna med denna uppdelning av kommandon är:

- Det blir lättare att underhålla eftersom varje kommando kan underhållas för sig. Man behöver inte uppdatera hela spelarobjektet för varje liten ändring.
- Hjälpen finns nära det kommando det handlar om. I min implementation hamnar det i samma fil och det går inte att lägga hjälptexten någon annanstans.

---

[1] The Mud Institute, ett mud i USA med målet att producera ett bra mudlib, även namnet på mudlibbet till skillnad från CD mudlibbet.

- Spelarobjektet innehåller funktioner för att sätta om sin path. Den beror nämligen av spelarens variabel `PATH`.
- I min implementation så kan man pipea saker som i unix mellan vissa av kommandona:

```
head *.c ^ grep filen ^ more
```

Problemet med det hela är att det inte är riktigt genomfört. Det finns bara ett begränsat antal kommandon som man kan kombinera på detta sätt.

Ett annat problem är att det är ganska svårt att skriva nya kommandon. En utmaning kanske? Läs filerna '`/bin/IDE`' och '`/bin/S]FUNKARDET`' för att se hur det är tänkt att det ska fungera.

4.1.2 '`doc`'

All dokumentation för hur spelet fungerar samlas här. Det kan vara bra att läsa igenom allt under denna katalog innan man börjar jobba med sina objekt. Viktigast är nog '`build`'dirret.

I flera av underkatalogerna finns det en katalog som heter '`obsolet`'. Där har jag flyttat ner de filer som ersatts av nyare, i samband med översättning eller omskrivning.

4.1.2.1 '`build`' - hjälpfiler för standardprylar

I denna katalog finns bland annat filen '`/doc/build/REGLER`'. Den innehåller lagar och förordningar som reglerar magikernas uppträdande och vad magikerna får göra och inte får göra.

Denna fil skall läsas av alla magiker!

Andra bra filer som du ska titta på är '`vapen.lista`', '`monster.lista`', '`skydd.lista`' och '`drycker.lista`'. De innehåller information om de olika objekten och riktlinjer för hur objekten ska se ut.

4.1.2.2 '`efun`' - funktioner som drivern erbjuder

De funktioner som finns i drivern brukar kallas efunar (engelska: external functions.). De är de grundläggande funktionerna och allt man egentligen har när det gäller att bygga sina objekt.

De flesta efunar finns beskrivna i denna katalog. Har du magikerboken kan du komma åt dem t.ex beskrivningen av `find_player` med '`man find_player`'. Jag har nyligen stoppat in en kopia Profezzorns '`lpc_for_morons`' här. Detta för att den är bättre.

De gamla filerna kan du hitta i '`/efun.old`'.

4.1.2.3 '`lfun`' - funktioner i standardobjekt

Funktioner i vissa standardobjekt kallas för lfunar (ursprungliga engelska betydelsen local functions).

De funktioner som beskrivs är främst de som finns i monster och prylar, men även en del viktiga funktioner som anropas från drivern:

- `catch_tell` anropas i objekt som gjort `enable_commands` om någon pratar till objektet, dvs objektet finns i ett rum där någon gör `say`, som någon gör `tell_room` till eller liknande.
- `init` anropas när ett objekt flyttas in i ett annat objekt. Dels anropas `init` i objektet som vi flyttar in i och dels anropas `init` i alla andra objekt som redan fanns i det objektet vi flyttas in i.
- `exit` ska du helst inte använda.[2]
- `id` anropas när man gör `present`. Den ska returnera sant eller falskt beroende på om objektet känns vid en visst namn.

4.1.2.4 '`exempel`'

I denna katalog har jag samlat exempel på kod. De flesta exempel rör hur man hanterar uppdrag, matobjekt och dessutom några

---

[2] Detta är en av de mest välbevarade myterna i LPmudvärlden. I drivrar av version 2 så anropas nämligen `exit` i rummet när man ska flytta ut ett objekt ur rummet. Om det blir ett fel i den funktionen fanns det ingen som helst möjlighet att få ut spelaren ur rummet. SvenskMUD kör en driver version 3 och där sker inte detta. Du kan alltså fritt använda funktionen `exit` som vilken annan funktion som helst.

som handlar om hur man uppdaterar objekt från COMPATmod till NATIVE. Här borde nog egentligen lagts ner mer arbete men jag föredrar att skriva denna bok. Om du har något bra exempel på någonting så lägger vi in det här för andra att lära sig av.

4.1.2.5 'uppdrag' - uppdragsbeskrivningar

Alla uppdrag ligger beskrivna här, med lösningar. Ett uppdrag är godkänt om det finns med i detta dir och inte godkänt om det inte finns med.

Detta dir ska vara lässkyddat för alla andra än SvenskMUDs magiker.

4.1.3 'etc'

Här sparas bland annat breven som posten skickar ut och breven som ligger och väntar på att du ska läsa dem.

4.1.4 'include'filer

Det finns en default-path för att hitta include-filer som används om includefilen refereras med

```
#include <filnamn>
```

Detta dir är först i denna default-path. Saker som ligger här är bl.a '`gender.h`' som används för att hantera könen och '`tune.h`' som reglerar hur spelet är justerat, dvs hur mycket ett erfarenhetspoäng kostar i ören.

4.1.5 'log' - loggfiler

Det finns en speciell efun som används för att logga saker nämligen `log_file`. Vilket objekt som helst får skriva med den men då hamnar det alltid i detta dir. Här finns det också vissa filer som skrivs hit av spelet.

Se till att rensa dina filer här regelbundet.

De magikerspecifika filerna för magikern '`linus`' är:

'`linus`' När du laddar objekt och får fel så skrivs felmeddelandena i denna fil. Det enklaste sättet att hantera den under programmeringsfasen är att:

1. radera filen
2. ladda sitt objekt
3. om det blir fel så tittar man i filen

'`linus.rep`'

När någon spelare eller magiker skriver `bugg`, `ide`, `stavfel` eller `beröm` så hamnar det i din denna fil om

- Han står i något av dina rum när han gör det.
- Han anger något objekt som mudden associerar till dig. (Detta är en ganska obskyr finess. Om man skriver '`bugg svärdet kan inte döda`' och `present` i spelaren hittar svärdet och du har gjort svärdet så hamnar det hos dig. Om spelaren menar ett annat svärd än det `present` hittar så hamnar det nog fel.)

4.1.6 '`obj`' - muddens grundobjekt

Här återfinns alla muddens standardobjekt i en salig blandning. Dels vissa specialobjekt med speciella funktioner ('`/obj/shut`', '`/obj/leo`') och dels objekt som man både kan ärva och klona ('`/obj/monster`', '`/obj/treasure`', '`/obj/weapon`' mfl.)

4.1.7 '`rum`' - muddens ursprungsvärld

Här ligger alla rum som ingår i den ordinarie världen. Det finns en uppdelning i underbibliotek för olika geografiska områden. ('`/rum/generisk`', '`/rum/|ster}ker`', '`/rum/strandhamn`' mfl.)

Av någon outgrundlig anledning (tradition heter det nog) ligger även filen '`init_file`' i detta dir. Det är den filen som läses när man ska avgöra vilka slott som ska läsas in när spelet startas om.

4.1.8 'secure'

Viktigare objekt ligger här.

Det viktigaste är masterobjektet. Masterobjektet har dels hand om säkerheten och dels reglerar det vad som händer när spelet bootar om.

Se kapitel 6 [Masterobjektet], sidan 27.

4.1.9 'spelare'

Under spelardirret sparas alla spelarnas data, dvs hur många poäng de skrapat ihop, hur lång tid de varit inne mm. Dessa spelarfiler skrivs med `save_object` på spelarobjektet när man gör `spara`, råkar ut för '`Jag sparar dig automatiskt.`' och när man gör `sluta`.

Dessutom är det här som magikerna får sin filkatalog som de själv bestämmer över. Denna skapas när man släpper sitt slott och då skrivs också slottet dit.

En speciell fil är '`castle.c`'. Denna laddas vid uppstart och är din enda möjlighet att länka in ditt område. Denna autoladdningsmöjlighet regleras i filen '`/rum/init_file`' och kopplas på när du släpper ditt slott. Den kan givetvis också plockas bort om du inte sköter ditt område eller inte vill ha det med i spelet längre.

4.1.10 'std'

En del standardobjekt, grundstommar för andra objekt, finns här. Bl.a '`/std/object.c`' som är det standardobjekt som alla saker som existerar i spelet ska ärva.

'`/std/namn.c`' är en sak jag gjorde när jag flyttade alla funktioner som hade med namn att göra från '`/obj/treasure`', '`/obj/weapon`' mfl. hit.

4.1.11 'stdobj' - enkla färdiga prylar

Ibland kan det vara så att man vill ha ett enkelt standardvapen eller en annan standardpryl. Det är tänkt att man ska kunna klona dem härur. Radagast har varit flitig och lagt in ett vapen av varje vapenklass samt dörrar.

I underkatalogen '`magiker`' finns magikerverktyg dvs. objekt som magiker kan använda som verktyg för sitt konstruerande.

4.1.12 '`texter`'

I '`texter`'-katalogen sparas vissa texter som används i mudden. Bl.a. texterna man får upp när man loggar in, '*SvenskMUDs Dagblad*' och topplistan över spelarna.

Det kan kanske vara intressant för magikerna att veta att alla gamla nummer av '*SvenskMUDs Dagblad*' och '`Svenska Mudbladet`' finns sparade här.

4.1.13 '`|vrigt`' gammalt skräp

En gammal katalog som ligger kvar. Det går inte att ladda in och använda objekt som ligger i denna katalog för det tillåts inte enligt masterobjektet. Om du har saker som borde ligga här så har du fel. De borde kanske ligga i '`/stdobj`'.

4.2 Driverns loggfiler

'`LP_SWAP.3.bodil`'
: För att spara minne slänger drivern ut programmen för de objekt som inte använts nyligen på denna fil. Där hämtas de när de behövs. Filen förlängs hela tiden och rensas bara när drivern startar om.

'`OBJ_DUMP`'
: Denna fil innehåller en lista av alla objekt i hela världen. Den uppdateras bara om man ger kommandot `dumpallobj` i muddet. Det kan ibland vara användbart för att få reda på om saker finns, vilka saker som finns och var saker finns.

'`lpmud.log`'
: Vi kör ett återstart-skript som fixar så att stdout och stderr från drivern hamnar i slutet av filen

'`lpmud.log`'. Det som drivern skriver på stdout är när den misslyckas att ladda in vissa objekt. Dessutom om man anropar `write` utan att `this_player` är satt till någonting så skrivs det ut på stdout (med ett `]` före).

'`bodil.debug.log`'
Detta är driverns egen loggfil. Dvs om det blir exekveringsfel i något objekt så skrivs det ut en stack backtrace här med funktionsnamn, radnummer och objekt.

5 Hur fungerar objekt?

Objekt i mudden associeras alltid med ett filnamn. Om filnamnet är '`/obj/hej.c`' så får objektet namnet '`/obj/hej`' och kloner till det objektet får namnen '`/obj/hej#123`' där '`123`' är ett nummer som tilldelas i sekvens från 0 då världen startar om.

Objektet med namnet '`/obj/hej`' kallas hädanefter för förlagan och objekten med namnen '`/obj/hej#123`' kallas för kloner eller klonade objekt.

För varje typ av objekt kan det antingen bara finnas kloner eller också bara förlagan inne i spelet. Om man försöker klona ett objekt där redan förlagan finns i spelet så får man '`Cloning a bad object`'. Om man försöker använda (ändra variabler i) förlagan för ett objekt som är klonat så får man '`Using a cloned object`'.

5.1 Förlagan

Om man vill vara säker på att det bara finns en enda kopia av en viss pryl i muddet ska man göra det som en förlaga.

För rum finns normalt bara en förlaga. För att göra om rum så gör man '`uppdatera` *filnamn*' och sedan går man in i dem igen eller laddar in dem med '`ladda` *filnamn*' .

Ditt slott ('`castle.c`') är också bara en förlaga. Det innebär att du inte ska klona det utan det ska laddas.

`create` anropas i förlagan när den laddas.

5.2 Kloner

För att skapa en klon av ett objekt använder man helt enkelt `clone_object` och man anger då som argument namnet på förlagan. Det man får tillbaka är en nyskapad klon.

```
object hej;

hej = clone_object("/obj/hej");
```

Om förlagan inte är laddad så laddas den först och `create` anropas i den, därefter klonas ett objekt och `create` anropas i klonen.

5.3 Vad finns i objektet?

Färdigt i objekten finns möjligheter att flytta objekt in i och ut ur varandra, möjligheter att ta reda på vilket objekt jag finns i samt vilka objekt som finns i mej.

Dessutom finns det möjlighet att avgöra om ett objekt är ”living” eller inte. Att det är ”living” innebär att det har ett namn så att man kan hitta det med `find_living`, som gör det snabbt att hitta monster och spelare.

Objekt kan också ha en s.k. ”heart_beat”. Det innebär att funktionen `heart_beat` anropas regelbundet i objektet. Alla monster använder sig av detta för att slåss och räkna upp sin egen ålder. Spelare använder den dessutom för att hela.

Om man kan så skall man försöka undvika att använda sig av `heart_beat` eftersom det anropas så ofta. I de allra flesta fall kan man klara sig lika bra med en `call_out` som har betydligt längre tidsintervall.

5.4 ’`std/object.c`’

Om man bara vill göra ett enkelt program så kan man kanske nöja sig med det men oftast vill man även ha möjlighet att flytta in objektet så att det fungerar i spelet. För att göra det måste objektet (i något led) ärva filen ’`/std/object.c`’.

Alla standardprylar som man ärver ifrån, ’`/rum/rum`’, ’`/obj/weapon`’, ’`/obj/treasure`’ mfl gör redan detta så om man använder någon av dem är det inga problem.

I ’`/std/object`’ finns det inte så mycket och egentligen borde det finnas ännu mindre där.

Det som finns är bl.a. en `move_object` som gör att man kan flytta vilket objekt som helst. Normalt, i NATIVE, är att varje objekt bara kan flytta sig själv med `move_object`.

Jag har slängt in genushanteringen här. Det innebär att alla objekt i spelet har funktionerna `set_maskulinum`, `set_femininum`, `set_utrum`, `set_neutrum`, `query_gender`, `query_pronoun` mfl och dessutom en variabel som håller reda på detta.

Om du inte ärver '`/std/object`' i något led så går det inte att flytta in objektet i spelet. Det existerar men finns utanför. Detta kan vara förvillande om man inte känner till det.

6 Masterobjektet

Masterobjektet har två uppgifter. Vid start av spelet så är det masterobjektet som bestämmer vad som ska göras och dels när det gäller vissa speciella efunar, de som skriver och läser filer. Masterobjektet är det objekt som får frågor från drivern om huruvida en viss läsning av en fil är tillåten eller inte.

6.1 Vid start

När spelet startar så sker följande:

- Funktionen `create` anropas i masterobjektet.
- Funktionen `flag` anropa en gång för varje argument till flaggan -f som drivern startas med.
- Funktionen `epilog` anropas.

  Det är i denna funktion som filen '`/rum/init_file`' läses och förladdar det som ska förladdas.
- Sedan börjar spelet lyssna efter uppkopplingar.

6.2 Säkerheten

Masterobjektets andra stora uppgift är säkerheten i spelet. Det finns ett antal `valid_`*whatever*-funktioner som anropas när man gör olika "farligare" saker. När man ska skriva på en fil så anropas `valid_write` för att kolla att du får lov att skriva på den filen. Den anropas med filnamn, vilket sätt man håller på att skriva filen med och vem som gör det. Sedan avgör funktionen om det är tillåtet eller inte.

Om du läser masterobjektets fil ser du att `valid_read` alltid returnerar sant. Det är ett medvetet val. Alla ska ha möjlighet att läsa överallt!

7 Standardgrejor

Det som man bör tänka på oavsett vad det är man bygger är att man ska få det att passa in i stilen och reagera på vissa standardkommandon. Ett par handskar behöver inte vara utrustade med en CRAY-17 armbandsdator med möjlighet att skicka laserstrålar och ett monster behöver inte springa omkring överallt och skapa 10000 kloner av sig själv. Oftast är det mycket roligare om man håller sig till stilen och gör välgenomtänkta saker med finesser som de som tänker lite kommer på. T.ex att man bara kan fiska om man har ett metspö, bara kan se vad som finns i kistan om man har öppnat den först.

7.1 Skatter

Den enklaste typen av objekt att bygga är nog skatter. Det är bara att klona '`/obj/treasure`' och sätta värden och namn mm.

Enda svårigheten är att veta vilka värden olika saker ska ha. Det är också en avvägningsfråga när man ska avgöra hur mycket man ska dela ut.

Saker som bara ligger och skräpar ska vara billiga, typiskt under 20 öre. De är inte onödiga bara för att de är så billiga utan de är till för att även nybörjare ska ha något att plocka upp och springa och sälja. Om det ligger en liten liten bärnsten vid stranden eller en vacker blomma på ängen som man kan sälja i affären så kommer det också att bidraga till stämningen.

Saker som stora monster bär på och som är extra belöning för att man har lyckats slå ihjäl det kan ju vara lite värdefullare. Likaså saker som ligger i svåråtkomliga rum.

7.2 Pengar

Pengar är lite speciella. När de finns i spelaren och i monster så är de bara ett värde i en variabel. Om man dör så skapas ett objekt som är pengar och som är värt ett visst belopp. Tar man det objektet försvinner det och pengarna läggs till de pengar man redan har.

Att pengarna alltid visas som daler och ören är tillagt efteråt. Det finns en färdig funktion för konverteringen. I filen '`/obj/libfun`' finns funktionen `print_money` som tar ett belopp i ören som argument och returnerar en sträng.

7.3 Lappar, anslag och skyltar

Det finns i spelet ett antal skyltar, lappar och anslag. Vissa hämtar datan de visar från yttre filer, se t.ex. topplistan som ligger i puben och '*SvenskMUDs Dagblad*' .

Andra saker att tänka på är att skyltar, tidningar mm ska man kunna `läs`a, likväl som man kan `titta på` dem. Dessa båda sätt att undersöka dem på behöver inte resultera i samma text men båda möjligheterna ska finnas.

7.4 Vapen

När man gör vapen ska man titta i '`/doc/build/vapen.lista`'. Den innehåller en lista av lämpliga nivåer på olika vapen för att man lätt ska se hur bra man ska göra det. Här är det väldigt viktigt att göra rimliga saker. En kniv är oftast mycket sämre att slåss med än ett spjut t.ex. Hur mycket de olika sakerna är värda är helt reglerat i listan.

Speciella saker att tänka på är att vapen med högre nivå än 17 plockas bort ur affären om någon skulle sälja dem och att vapen med en högre nivå än 20 ska ha allvarliga nackdelar. Hur allvarliga de behöver vara bedöms av den ärkemagiker som godkänner det.

7.5 Rustningar och kläder

När man gör rustningar och kläder så finns det vissa saker som man måste tänka på för att alla rimlighetskrav ska uppfyllas.

Dels finns det i '`/doc/build/skydd.lista`' en lista över vilka nivåer som är tillåtna för olika rustningstyper och vilka kostnader de får ha. Dessutom ska man se till att hålla sig till de typer som finns, bokstavsgrannt, så att man inte kan ta på sig för mycket skydd.

Enklaste sättet är att klona '`/obj/armour`' och sätta namn, typ och klass.

7.6 Monster

När du bygger monster måste du se till att de blir "living", det blir de om du anropar `create` i '`/obj/monster`' (eller på något annat sätt gör `set_living_name` och `enable_commands`.

Även när det gäller monster är det viktigt att man använder sin kreativitet för att de ska bli trevliga. Monstren är inte automatiskt bättre för att de är starkare och man behöver inte börja med jättemonstret Allan. Även små monster är befogade för att ge atmosfär och ge tips för nybörjarna.

Glöm inte att sätta kön.

När ett monster dör skapas ett lik. Man kan i monstret sätta ett värde på hur mycket liket skall väga. Gör man inte det så väger liket 5, som ungefär motsvarar en människa. Mitt förslag är att sätta mindre (1-2) på katter och hundar och mer (8-15) på kor, hästar och björnar.

7.6.1 Speciell hantering av `heart_beat` i monster

Monster helas inte i `heart_beat` utan istället i funktionen `heal_slowly`. Detta beror på att när du lämnar ett monster så stängs `heart_beat` av för att slås på igen när någon kommer. `heal_slowly` anropas via `call_out` varannan minut ända till monstret är helt läkt. Då slutar den för att startas igen nästa gång någon börjar slå på monstret. Detta sparar kraft för drivern i och med att monstrets `heart_beat` stängs av.

Om du gör ett monster så se till att försöka använda denna eller liknande mekanism för att slå av `heart_beat` då monstret står ensamt i ett rum.

En annan sak som bör observeras är att monster inte har någon magisk styrka. Hur många besvärjelser de kastar beror helt och hållet på vilka slumpfaktorer du har satt.

7.6.2 Monster av olika raser

Monster har en parameter som man kan sätta som avgör vilken ras monstret är av. Den ska sättas till ett ord i obestämd form singularis bestående enbart av gemener. Exempel. hund, katt, troll, människa.

Egentligen borde det finnas färdiga monster att klona fram och bara sätta nivån hos. Det borde i så fall ligga i '`/stdobj/monster`'. Då borde det finnas en fil/objekttyp för varje ras.

Exempel på vad man skulle kunna göra är:

insekt
: Speciella egenskaper: lämnar inget lik, har väldigt hög smidighet, smiter lätt in i ett annat rum. Man sätter namnet (mygga, fluga, bi, geting, broms).

fågel
: Speciella egenskaper: lämnar ett lätt och litet lik som är ätbart, har hög smidighet. Man sätter namnet (koltrast, stare, blåmes, kråka, skata, berguv...), nivå.

gnagare
: Speciella egenskaper: ganska snabba, bits. Man sätter namnet (bäver, sork, ekorre, råtta, mus...), om det är ett vatten-, mark- eller träddjur.

generiskt däggdjur
: Speciella egenskaper: lämnar ett ätbart lik, ganska tungt, slår ganska hårt. Man sätter sorten (ko, häst, älg, ren) och namnet (Rosa, Brunte, Hälge, Rudolf), storlek/styrka.

troll
: Speciella egenskaper: förstenas om de kommer ut i dagsljus, har svans, är aggressiva. Man sätter styrkan, vilket vapen det ska ha, namnet, deras texter.

7.7 Rum

Speciella saker att tänka på när man gör rum är att göra beskrivningar för olika saker. Har man en pub kan det vara kul om man kan titta på bardisken eller något och få en liten text om den.

I det vanliga '`/rum/rum`' så kan man enkelt klara detta med `items`.

Andra saker att tänka på är att göra vettiga beskrivningar och att inte överlasta rummen med prylar.

Om man använder sig av ordinarie rummen så kommer de att rensas bort vid `clean_up` om de är tomma. Dvs efter ca 2 timmar om ingen har varit i dem. Vill man behålla dem, för att man lagrar variabler eller annat där, så får man definiera om `clean_up`.

7.8 Inflation i spelet

Inflation uppkommer om magikerna utan begränsning skapar dyrare och dyrare eller bättre och bättre saker.

Nackdelarna är att nybörjare får det svårare (dvs även de enklaste monster är för svåra) och att man måste göra om kraven hela tiden, kanske ändra till nivå 30 för att bli magiker med 100 miljoner erfarenhetspoäng.

De mekanismer som finns för att motverka detta finns där för att motverka detta och inte för att kringås!

- Affärer betalar bara upp till tio daler för prylar, aldrig mer.

  Se till att inte göra saker som är dyrare än tio daler, det retar bara upp spelarna.
- Listorna för vapen, rustningar och monster ska följas. Att göra dyrare och bättre saker hela tiden är ett symptom på att man är ny som magiker eller att man har dålig fantasi. Det är också den främsta anledningen till att det blir inflation.

8 Uppdrag

När man kommit en bit på väg med sitt byggande är det kanske dags att börja fundera på att göra ett uppdrag.

Uppdragen har tre syften:

1. De ska stimulera spelarens upptäckarglädje och uppmuntra till tankeverksamhet.
2. De ska hindra att spelandet blir mekaniskt, dvs se till att man inte kan spela enbart mha. makron.
3. De ska tvinga spelarna att utforska väldigt stor del av världen innan de blir magiker så att de vet vad som finns och gör sig en bild av världen.

8.1 Ditt mål med uppdraget

Det första du ska tänka ut när du börjar planera ett uppdrag är vad ditt syfte är med uppdraget.

Om ditt syfte är att spelaren ska utforska hela din del av världen så är det första du ska göra att se till att din del av världen går att gå i och "tål" att utforska. Dvs det finns något kul[1] i varje rum, något monster, någon pryl, något ihåligt träd, ...

8.2 Presentation av uppdraget

Du måste dikta ihop en liten historia om ditt uppdrag. Någon behöver hjälp med något är det typiska scenariot.

Eftersom uppdraget ska lösas flera gånger, en gång för varje magikeraspirant, så är det bra om du kan få ihop historien så att det på något sätt är förklarligt att det kan lösas flera gånger. Kanske ett litet skådespel en stund efter (medan spelaren står och begrundar belöningen) som på något sätt återställer saker så att det kan lösas på nytt.

---

[1] med kul menas här givetvis kul att undersöka, intressant beskrivning, intressanta fraser, intressanta effekter när monstret dör, när man tar på prylen eller... Inte att monstret eller prylen är starkast eller bäst i hela världen.

Appendix A Ordförklaringar

mud
: SvenskMUD är ett mud. Mud är en engelsk förkortning och står för Multi User Dungeon, (FlerAnvändarGrotta i direktöversättning). Vissa använder även begreppet mud för att beteckna spel som jobbar med ett grafiskt gränssnitt mot användaren.

klient
: Program som man använder för att koppla upp sig mot drivern. Den mest spridda är nog telnet.

driver
: Det program som kör själva spelet. Det tar hand om uppkopplingar från spelare samt laddar och interpreterar ("kör") filer.

LPC
: Det programmeringsspråk som man skriver objekten i i SvenskMUD och alla andra LPmud. Det liknar C.

LPmud
: Muddar som arbetar enligt samma princip som Lars Pensjös ursprungliga mud, Genesis. Dvs man skriver objekten i LPC.

mudlib
: Förutom drivern består själva spelet också av en massa objekt. Dessa utgör mudlibbet. Ibland när man pratar om mudlib av olika versioner så brukar man dock inte inkludera de enskilda magikernas filer.

magiker

dödliga
: När man börjar spela är man en dödlig spelare, men om man klarat alla uppdrag och samlat ihop tillräckligt med erfarenhet blir man upphöjd till magiker. Det innebär att man får lov att modifiera världen genom att skapa nya saker.

: Man kan betrakta magikerskapet som målet med spelet och det är det enda mål som finns i ett LPmud, förutom att ha roligt.

: I vissa delar av den här boken har jag kallat magikerna för författare eller skapare. Det är de som skapar spelet och utan dem vore spelet dött.

ärkemagiker
: En ärkemagiker är förmer än andra magiker. De är de som övervakar magikernas arbete och bestämmer över världen.

gud Gud är självutnämnd. Det visar att jag är förmer än andra magiker och ärkemagiker. Om du undrar varför det är på detta viset så beror det på att det inte finns någon fungerande demokrati i SvenskMUD. Om det inte finns någon fungerande demokrati är de enda alternativen diktatur eller kaos.

Det kan bara finnas en gud.

efun external function. Dvs. en funktion som finns i drivern. De är implementerade i C och därför effektivare. Dessa kan användas från alla objekt.

lfun local function. Dvs. en funktion som finns i någon fil och som anropas antingen från drivern, från ett annat objekt eller från samma objekt.

Dessa är skrivna i LPC och exekveras därför mindre effektivt än efunarna.

simul_efun En funktion som fungerar precis som en efun men som inte är en efun. De finns definierade i filen '`/obj/simul_efun.c`' och det är fixat så att de kan användas från alla objekt utan att man behöver ärva något speciellt.

Dessa är också skrivna i LPC.

förlaga För alla objekt i spelet finns en förlaga. För vissa objekt, t.ex rum, används bara förlagan.

kloner När man har en förlaga kan man skapa kloner till denna. De är exakta kopior så när som på att de har en egen uppsättning variabelvärden.

Appendix B Vad skiljer C och LPC

Tack till: *Erik A. Kay – Wayfarer* `@` *TMI* som jag lånat det mesta av detta kapitel av.

B.1 Vad har C som inte LPC har?

1. Stark typkontroll.

   I LPC finns ingen egentlig kontroll att man i en variabel av en viss typ verkligen har ett värde av den typen. Man kan däremot deklarera funktioner så att de tar vissa typer som argument och vid kompileringstillfället kolla att alla anrop är korrekta.

2. Typerna `unsigned`, `float`, `double`, `long`, `char`, `void *` mfl.

   Vissa av dessa ersätts helt eller delvis av andra typer.

3. Konstruktionerna `struct`, `union`, `enum`, `typedef`.

   Över huvud taget existerar inte alls mycket konstruktioner som ska beräknas vid kompileringstillfället.

   `struct` och `union` kan man implementera med mappings eller arrayer men hela hanteringen måste ske vid exekveringstillfället. `enum` och `typedef` tvingas man emulera med `#define`.

4. C's standardbibliotekfunktioner

   Det går inte att komma åt C's biblioteksfunktioner från de olika objekten. Vad man kan göra och inte kan göra är begränsat av vilka efunar som finns. Man tvingas lägga till nya efunar om man vill utöka detta, det innebär en omkompilering av drivern.

5. Statiska lokala variabler

   Man tvingas göra en objekt-global variabel.

6. Globala variabler

   Man tvingas göra ett speciellt objekt hos vilket man registrerar variabelvärden. Personligen tycker jag detta är en snygg lösning.

7. Flyttal

B.2 Vad har LPC som inte C har?

Dessa skillnader kan delas in i tre stora grupper:

- Saker som sker vid körningen av ett objekt.
- Saker som har med objektorienteringen att göra
- Strängar och arrayer och speciella funktioner för dessa.

B.2.1 Vid körning av kod

Alla objekt är av en och samma typ. Det innebär att funktionsanrop i objekt måste vara väldigt generaliserad. Det finns egentligen bara ett enda sätt att anropa en funktion i ett annat objekt och det är med funktionen `call_other`. Den tar som argument ett objekt eller möjligtvis filnamn, ett funktionsnamn och eventuella argument. Argumenten kan vara av godtycklig typ och det finns inget som helst sätt att vid kompileringen av objekt, kolla om man har rätt typer.

Vidare returnerar `call_other` någonting som har datatypen `mixed` så ska man använda det vidare måste man tala om vad det är för typ, med ett c-liknande `cast`.

B.2.2 Objektorienterade funktioner

- Anropandet av funktioner i andra objekt. Man kan anropa funktioner i andra objekt med `call_other(`objekt, funktionsnamn, argument`)`[1].
- Multipel ärvning.

  Det är inga som helst konstigheter med att ärva flera olika filer. Man kan också själv välja från vilket ärvt objekt som en funktion ska anropas. Ex från facklan:

  ```
  inherit "/std/object";
  inherit "/std/namn";

  short()
  {
  ```

[1] Man kan även skriva: objekt`->`funktion`(`argument`)`

```
        if (is_lit)
            return namn::short() + " (tänd)";
        else
            return namn::short();
    }
```

Det innebär att `short` i '`namn`' anropas även om det skulle finnas en `short` i '`/std/object.c`'.

- Variabler och funktioner kan vara privata.

  Om man vill att de ärvda objekten inte ska kunna modifiera en variabel mer än genom vissa funktioner kan man göra variabeln privat (`private`). På samma sätt kan man begränsa funktioners räckvidd.

- Funktioner kan göras unika.

  En funktion kan deklareras så att inget objekt tillåts definiera om funktionen efter att ha ärvt det objektet. Det gör man med uttrycket `nomask`.

B.2.3 Strängar, arrayer och mappings.

Stränghanteringen i LPC är väldigt enkel. När man summerar strängar så slås de ihop till en lång sträng. Blandar man in tal så konverteras de till strängar först. För att gå från en sträng som innehåller ett tal till ett tal använder man `sscanf`.[2]

Arrayer kan man också addera och då får man en ny array som är sammanslagningen av de två. Man kan också subtrahera arrayer från varandra, det innebär att man får en ny array som innehåller alla element i den första arrayen utom de som finns i båda.

På både strängar och arrayer kan man ta ut delar av dem med hjälp av intervall-operationer. Ex. '`"hej"[1..2]`' blir '`"ej"`'.

---

[2] Ej att förväxla med C-bibliotekens `sscanf`, den som finns här är mycket enklare och är en del i språket. Man ska exempelvis inte skicka med pekare till tal och strängar utan variablerna direkt.

Mappings är en helt ny typ. Man kan kalla det en slags array där indexen kan vara vad som helst. Dessutom finns det funktioner för att ta fram alla index eller alla värden.

Appendix C Problemsökning

Ofta ställda frågor med svar eller några ideer på vad som kan vara fel.

När jag lade till min gömda utgång i mitt rum slutade riktningarna definierade med dest_dir att fungera.

Du har definierat om `init()` (för att lägga till dina saker). För att `dest_dir`-riktningarna ska fungerar måste du anropa `rum::init()` i `init()` så att det görs `add_action()` på `dest_dir`-riktningarna.

Efter att jag lade till min finess i `heart_beat()` slutar monstret att slå.

Se till att du anropar `::heart_beart()` från `heart_beat()`. Den sköter stridsmekanismen samt ser till att slå av heart_beat när ingen spelare finns i närheten (sparar mycket exekveringstid för drivern).

Den slår på mig men om jag går ut ur rummet och väntar två sekunder så slår den inte igen.

Har du definierat om `init()` för monstret? Om du glömt att anropa `::init()` så kommer nämligen inte heart_beat att sättas igång automatiskt när någon kommer in.

När jag lade till en extra finess att utföras i `heart_beat()` på mitt monster så stängs heart_beat av hela tiden.

Du har antagligen introducerat något exekveringsfel. Kolla slutet av filen '`/bodil.debug.log`' och se om du kan hitta felet.

Jag har en pryl som ska komma fram varje reset men från början finns de inte där.

Anropar du `reset()` från `create()`?

Funkar det att klona dem vid sidan om?

Är de synliga? Om du sätter en pryls namn mm i `reset()` istället för i `create()` så blir de osynliga fram till första reset.

När jag klonar fram mitt objekt som ska generera ett annat objekt (paret) så får jag ibland två andra objekt.

Du skapar paret från `create()`. `create()` anropas först i förlagan och sedan i klonen. Kanske det ena paret är pare till förlagan.

Jag har gjort en specialpryl men den "finns inte", dvs jag lyckas inte klona den trots att den laddar utan fel.

Har du ärvt '`/std/object`' i något led?

Appendix D Magikerverktyg

Magikerverktyg kallar jag de saker som magiker bär omkring på och som de använder i sitt arbete. De är oftast gömda under något till synes oskyldigt namn men kan ha riktigt kraftiga egenskaper.

Det finns magiker som blir besatta av magikerverktyg och som koncentrerar hela sin verksamhet på att göra bättre och bättre verktyg hela tiden. Låt inte det smitta av sig, du behöver inte också göra ett magikerverktyg! Utnyttja de som redan finns!

D.1 Magikerboken

Det enklaste magikerverktyget och förhoppningsvis det första du stöter på är magikerboken[1]. Det är som det låter en liten bok där det står lite om olika saker som man kan ha nytta av. Dessutom definierar den kommandot `man` som slår upp hjälptexter. Den söker igenom vissa kataloger efter filer som matchar ett visst namn när man anger det.

Du får magikerboken automatiskt när du blir nivå 20. Annars finns den i '`/obj/magikerbok`'.

Magikerboken definierar följande kommandon:

- `läs sida` *nr*
  Visar en sida ur magikerboken. Sida 1 innehåller en kapitellista med sidnummer.
- `man` *funktion*
  Slå upp hjälpen om funktionen *funktion* ifall det finns någon.
- `MANPATH` Visar vilken sökväg som är inställd.
- `MANPATH+=`*bibliotek*
  Lägg till ett bibliotek i sökvägen för `man`-kommandot.
- `MANPATH-=`*bibliotek*
  Ta bort ett bibliotek ur sökvägen för `man`-kommandot.
- `MANPATH=default`
  Sätt tillbaka MANPATH till det fördefinierade värdet.

---

[1] Ej att förväxla med den här handboken.

D.2 Ekstaven - Fizbans stav

Fizbans stav eller ekstaven är mer avancerad. Den ger möjlighet att anropa godtycklig funktion i godtyckligt objekt. De speciella ekstavsfunktionerna börjar oftast med stor bokstav.

När man skall ange objekt till staven kan man antingen ange filnamn (för kloner följt av `#`*nummer*), en spelares namn eller ett objekt som finns i närheten.

Dessutom kan man ange *objekt*, *n* vilket betyder det *n*:te objektet i *objekt* eller också *objekt*, *namn*. Då söker den igenom *objekt* efter det objekt som bekänner sig vid namnet *namn*.

Några av ekstavens kommandon är:

- `Hjälp` Stavens hjälptext. Allt man behöver veta om staven skall stå där.
- `Ä` Kollar vad jag äger på stavens vis.
- `Kolla` *objekt*
  Undersöker någonting på stavens vis.
- `Titta` Undersöker rummet på stavens vis.
- `Beskriv` Beskriver rummets egenskaper.
- `testa` *objekt*
  Testar ett vapen, skydd eller monster. Denna funktion är avstämd mot listorna '`/doc/build/*.lista`' och ger varningar när prylarna inte uppfyller kraven.
- `anropa` *objekt funktionsnamn argument*
  Anropa funktionen *funktionsnamn* i objektet *objekt*.

Exempelvis:

```
> Ä
Objekt i OBJ(obj/player#2741) <-> Linus skaparen
(gud)
 0: OBJ(stdobj/magiker/brain#2747) <-> linus' shell
 1: OBJ(obj/magobj#2750) <-> Osynligt objekt.
 2: OBJ(obj/magikerbok#2749) <-> en magikerbok
 3: OBJ(stdobj/magiker/fizbans_stav#2748) <-> En
gammal ekstav
 4: OBJ(obj/soul#2742) <-> Osynligt objekt.
Det var allt.
```

```
> anropa mej,staven long
Du inser att staven har en magisk kraft.
Returvärde : 0
> anropa mej,2 short
Returvärde : en magikerbok
>
```

Staven hittar du i '`/stdobj/magiker/fizbans_stav`'.

D.3 Shellen - brain

Profezzorn (i nannymud) har gjort en sak som han kallar shellen. Den kan användas av både magiker och spelare men den är inte översatt till svenska så jag har fixat så att spelare blir av med den hela tiden.

Mha den kan man göra en del saker direkt som man annars skulle behövt ett speciellt objekt för.

Objekt kan anges på samma sätt som med ekstaven.

De mest användbara av shellens kommandon är:

- `shellhelp wizard`
  Skriver ut en hjälpsida. Man kan få hjälp på de olika rubrikerna genom att skriva `shellhelp` *rubrik* .
- `id` Man kan få fram information om när nästa reset inträffar, hur stort programblocket är, hur mycket minne variablerna tar mm.
- `gauge` *kommando*
  Ger information om hur arbetskrävande ett kommando är.
- `eval` *uttryck*
  Evaluera ett godtyckligt lpc-uttryck.

Du kan klona shellen ur '`/stdobj/magiker/brain`'.

Register

/

A

B

C

D

M

N

O

P

Q

R

S

T

U

V

|

å

Ä

Innehåll