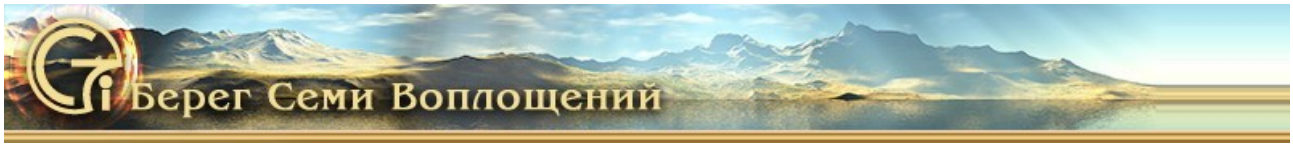

Руководство разработчика игрового MUD-сервера

«Берег семи воплощений».

Описание скриптов для языка Lua.

Редакция 1.005

2011-2016

Оглавление

Введение

Статичность и однообразие поведения мобов

Многие из игроков наверное задумывался - почему в большинстве своем мобы ничего не делают, а только сидят и ждут, когда кто-нибудь придет их убить? Почему, например, моб-убийца не ищет сам свою потенциальную жертву? Почему преступник, убить которого дали задание, стоит на месте и ждет своего палача, как баран мясника? А не делают они ничего, потому что никто не знает, что именно должен делать конкретный моб или объект и когда. Задача же реализации искусственного интеллекта слишком сложна, и вряд ли под силу кому-нибудь в ближайшее время.

Попытки разнообразить мир

Один из путей заключается в написании так называемых spec_procedures, подпрограмм, являющихся частью кода сервера MUDa, написанных на C (или C++) и вызываемых раз в секунду. Этот путь имеет несколько серьезных недостатков, например, необходимы специфические и достаточно глубокие знания C (или C++) и доступ к основному коду сервера. Для исправления любой ошибки, параметра или даже опечатки в сообщении необходима полная перекомпиляция сервера, . Поэтому данный путь является тупиковым. Ещё один аргумент — серьёзная ошибка в коде сервера вызывает останов сервера и перезагрузку (такая ошибка в терминах mud называется – crush bug – краш -баг).

Язык Lua

В настоящее время к нашему игровому серверу подключен язык Lua, независимый от основного кода, который определяет набор событий и реакции на эти события. Он достаточно простой, чтобы для его использования не требовались глубокие знания программирования, и в тоже время он достаточно мощный для реализации практически всего, что может потребоваться при работе с ариями игрового сервера. Также он поддерживает реализацию участков кода с большой загрузкой (использованием процессорного времени) на языке C++ и прямым вызовам этих функций из Lua.

В принципе, ничего не мешает подключить реализацию любого другого подходящего по условиям скриптового языка, хотя Lua получил признание и зарекомендовал себя как простой и удобный скрипт-язык во многих известнейших играх.

События

Ориентированность скриптов на события

Все скрипты, написанные для сервера ориентированы на события. Событием можно назвать практически любое происходящее в MUD действие. Например, персонаж ушел на север - это цепь событий: первое - ввод существующей команды движения (COMMAND), второе - уход персонажа из одной комнаты (EXIT), третье - событие (которое можно отменить) предваряющее его появление в другой (ENTER) , и четвёртое - появление в другой комнате (GREET), которое отменить уже невозможно.

Другие примеры событий:

- выполнение персонажем определенного действия, или выражение эмоции SOCIAL;
- периодически: раз в минуту (TICK), каждые 3-4 секунды (RANDOM), каждые 3 секунды - процесс битвы (FIGHT);
- начало битвы (ATTACK), смерть персонажа (DEATH), создание предмета или персонажа (INIT);
- передача ожидающему события персонажу предмета или денег - подкуп (BRIBE);
- выполнение команды или заклинания, которые реально могут даже не существовать (COMMAND, NEWCMD, CASTSPELL, CASTNEW);
- общение (SAY);
- использование объекта, который ждёт этого события (FROMOBJ, GET, GIVE, ONOBJ, PUT, REMOVE, USE, WEAR);
- совершение действия в комнате, которая ждет событие (COMMAND, FIGHT, NEWCMD, RESET, SAY и т.д.).

В данный момент список поддерживаемых событий вполне достаточен, но если будет необходимо, то его легко расширить.

Структура реакции на событие

Реакция на событие состоит из двух логических частей: объявления и реализации. Обе данные части описываются в файле арии. Реализация события может быть привязано к нескольким разным (не однотипным) событиям, например, событие TICK и INIT можно обрабатывать в одной реализации. Более того, можно делать реализацию для событий с разным количеством входящих параметров (недостающие параметры будут равны nil).

Объявление события: параметры и вероятность

Каждое событие тесно связано с объектом, который будет тем или иным образом реагировать на него. Существуют четыре типа объектов "получателей" событий: арии, комнаты, мобы и предметы. События вызываются непосредственно при наступлении обрабатываемого действия, при условии совпадения параметра и вероятности (если они указаны).

Формат объявления события имеет следующий вид:

event <прототип> lua:<функция-обработчик>() [CHANCE <знач>] [PARAM <знач>]

- event - обязательное ключевое слово, обозначающее, что далее в тексте следует объявление события.
- <прототип> - обязательное имя прототипа функции-обработчика события.
- lua: - указание на используемый скрипт-язык, сейчас таковым является язык Lua.
- <функция-обработчик> - обязательное имя вашей функции-обработчика, которая будет обрабатывать событие.
- () - необязательные символы, обработчик события будет найден и без этих скобок.
- CHANCE <знач> - опциональное значение вероятности срабатывания обработчика события после его возникновения. По умолчанию всегда 100%.
- PARAM <знач> - опциональное значения параметра, совпадение с которым будет проверяться при возникновении события. Указывается, если у прототипа функции-обработчика есть дополнительный параметр. Значение "all" (по-умолчанию, если PARAM не указан) означает, что функция-обработчик будет вызываться для любого объекта/действия.

Будьте внимательны: если внутри обработчика сделать некие действия, то можно вызвать срабатывание этого же обработчика. Например, внутри обработчика события SAY выполнив команду Char.execute(“say Вложенный цикл“) - это же событие будет вызвано многократно. Или внутри обработчика повреждений DAMAGE нанести повреждения Char.damage(1234), что опять вызовет бесконечный цикл. Нужно предусмотреть запрет/прерывание таких повторных вызовов.

Возвращаемые событием значения

Возвращаемое из обработчиков событий значения:

1. если не возвращается (нет команды return xxx), то результат считается 0 (обрабатывать дальше) [по умолчанию]
2. если return nil, то считается что результат 0 (обрабатывать дальше)
3. если return 0, то считается что результат 0 (обрабатывать дальше)
4. если return любое_число_отличное_от_нуля, то значит что последующая обработка будет прекращена
5. если return false, то считается что результат 0 (обрабатывать дальше)
6. если return true, то считается что результат 1 (последующую обработку прекратить)

Самая быстрая обработка в коде сервера - это пункт 1. Можно не писать return 0, return nil, код без этих команд будет выполняться чуточку быстрее и на вызов return не тратится память и процессорное время интерпретатора Lua.

Реализация события

Итак, вы объявили событие, то есть:

- выбрали объект, который будет на него реагировать (ария, комната, моб или предмет);
- задали корректное имя события (прототип функции-обработчика), который поддерживается выбранным вами объектом;
- придумали имя вашей функции-обработчика, которая собственно и реализует событие;
- определили шансы и параметры события.

Теперь вам необходимо его реализовать на языке Lua в блоке #LUA …. #~ любой арии, необязательно в той, где оно объявлено. Перед самой функцией напишите выделенный комментарий, в котором кратко опишите назначение обработчика. Это позволит другим кодерам (да и Вам впоследствии) быстро выяснить для чего предназначена функция:

```
-- ----------------------------------------------------------------

-- Вспомогательная функция. Укажите, если предмет (ЛИМИТ). Ну или другую краткую и понятную информацию.

-- ----------------------------------------------------------------
```

Однотипные события для разных классов Area, Room, Obj, Char, собранные в одном месте

Первый параметр функции func_name всегда содержит объект, обрабатывающий это событие (Area area, Room room, Char ch(mob), Obj obj). В большинстве случаев все отличия однотипных событий в этом и заключаются. Если событие класса:

- Area area - причина вызова произошла в любой комнате арии.
- Room room- причина вызова произошла в этой комнате.
- Obj obj- причина произошла рядом с обрабатывающим событие объектом (в комнате/инвентаре/экипировке) или с ним самим.
- Char ch(или mob, если в строке вызова упоминается два объекта Char) - причина произошла рядом с мобом/персонажем .

Параметр отбора PARAM стоит указывать для ограничения диапазона срабатывания. Для существующих команд, социалов, скиллов и заклинаний нужно указывать полное английское имя в нижнем регистре: "acid blast", а не "Ac bl", "charge", а не "Чардж". По-умолчанию, никаких ограничений нет. Если по каким-то причинам нужно ввести параметр «без ограничений», то указывают ”all”.

Отменяемые и не отменяемые события.
Некоторые события могут возвращать ненулевое значение (например, return 1) в качестве признака отмены действия, вызвавшего событие (т. е. запрет на дальнейшее выполнение команды для COMMAND, запрет входа в комнату для ENTER и т. д.) и всех последующих обработчиков событий для этого объекта! То есть, если есть два вызова COMMAND, то после отмены действия в первом обработчике второй не будет выполняться!
По умолчанию (если ничего не указывать),статус возврата равен нулю (разрешить выполнить действие). То же самое можно принудительно выполнить командой return 0.
Так как при отмене действия, вызвавшего событие ничего не сообщается, то стоит вывести персонажу (и/или окружающим) причину неисполнения действия.

Тип события	Параметры вызова обработчика	PARAM	Условие срабатывания
ATTACK разовое отменяемое	Room: func(Room room, Char victim, Char ch) Obj: func(Obj obj, Char victim, Char ch) Char: func(Char victim, Char ch) *ch - персонаж атакующий другого персонажа victim (в комнате room)* *obj - вещь у атакующего (obj.carried_by == ch) или атакуемого (не на полу!)*	Нет	У атакуемого/атакующего персонажей имеется вещи, обрабатывающий данное событие. Если одинаковых вещей N штук, то событие вызовется N раз.
DROP разовое отменяемое	Room: func(Room room, Char ch, Obj obj) Obj: func(Obj obj, Char ch) *ch - персонаж, бросающий вещь obj (в комнате room)*	Нет (TODO - сделать vnum?	Персонаж пытается бросить предмет в комнате.
CASTSPELL разовое отменяемое	Room: func(Room room, Char ch, строка spell, строка arg, число level, Char victim, Obj victobj) Obj: func(Obj obj, Char ch, строка spell, строка arg, число level, Char victim, Obj victobj) Char: func(Char mob, Char ch, строка spell, строка arg, число level, Char victim, Obj victobj) *ch - персонаж, колдующий существующее заклинание* *spell - полное (всегда) английское название заклинания* *arg - строка с аргументами заклинания* *level - уровень, с которым колдуется заклинание* *victim, victobj - персонаж, вещь - цель (или nil)*	Имя заклинания	Персонаж хочет сколдовать **существующее** заклинание рядом или в объекте, где есть обработчик данного события, и прочие условия для его исполнения (безопасность - safe, наличие жертвы, мана) выполнены.
CASTNEW разовое отменяемое	Room: func(Room room, Char ch, строка spell, строка arg, число level) Obj: func(Obj obj, Char ch, строка spell, строка arg, число level) Char: func(Char mob, Char ch, строка spell, строка arg, число level) *ch - персонаж, колдующий несуществующее заклинание* *spell - полный шаблон заклинания (англ. и русский нужно указать отдельно)* *arg - строка с аргументами заклинания* *level - уровень, с которым колдуется заклинание*	Полный строковый шаблон	Персонаж хочет сколдовать **несуществующее** заклинание рядом или несёт вещь, где есть обработчик данного события, Если не отменить действие, персонажу будет выдана ошибка "неверное заклинание".
COMMAND разовое отменяемое	Area: func_name(Room room, Char ch, строка cmd, строка arg) Room: func_name(Room room, Char ch, строка cmd, строка arg) Obj: func_name(Obj obj, Char ch, строка cmd, строка arg) Char: func_name(Char mob, Char ch, строка cmd, строка arg) *ch - персонаж, колдующий заклинание* *spell - полное английское название команды* *arg - строка аргументов после команды*	Имя команды	Выполнение **существующей** команды персонажем и прочие условия для её исполнения (мин. уровень персонажа, позиция) выполнены, если персонаж рядом или несёт вещь, у которого есть обработчик данного события.
NEWCMD разовое отменяемое	Area: func(Room room, Char ch, строка cmd, строка arg) Room: func(Room room, Char ch, строка cmd, строка arg) Obj: func(Obj obj, Char ch, строка cmd, строка arg) Char: func(Char mob, Char ch, строка cmd, строка arg) *ch - персонаж, отдавший несуществующую команду* *cmd - полный шаблон команды (англ. и русский нужно указать отдельно)* *arg - строка аргументов после команды*	Полный строковый шаблон	Выполнение **несуществующей** команды персонажем, если он рядом или несёт вещь, у которого есть обработчик данного события.
ENTER EXIT разовые отменяемые GREET(дубль ENTER для объектов Char!!!) разовое не отменяемое	(ENTER) Area: func(**Room room**, Char ch, строка dir) - *ария -> room.area* (ENTER,EXIT) Room: func(Room room, Char ch, строка dir) - *ch.room (откуда)* (GREET,EXIT) Char: func(Char mob, Char ch, строка dir) *room - комната, в которую пытаются попасть или уйти* *mob - персонаж, обрабатывающий событие* *ch - персонаж, пытающийся попасть/удалится из ch.room (откуда)* *dir- направление или действие с которого с которого совершается попытка попасть или удалиться. Принимает значения:* ***вход игрока в игру : entergame.*** ***заклинания: appearance, convoke, face to face, gate, group recall, hole, holy gate, summon, teleport, tornado, word of recall, word of transportation (fear и turn undead вызывают «Перемещение» в соседние комнаты!)*** ***перемещение: north, south, west, east, up, down, portal***	Ожидаемые направления или способы входа/выхода (“all” - для любых способов)	Персонаж пытается каким-либо способом попасть в комнату или покинуть её. В событии GREET - персонаж УЖЕ в комнате, куда он переместился, отменить вход невозможно! Если отменить действие для ”entergame”, игрок, заходящий в игру будет перенесён к алтарю. TODO: проверить все вызовы ENTER у заклинаний. Сделать quitgame для игрока
FIGHT периодическое не отменяемое	Room: func(Room room, Char ch, Char victim) Obj: func(Obj obj, Char ch, Char victim) - *obj - вещь у персонажа ch* Char: func(Char ch, Char victim) - *событие вызывается для ch* *ch - персонаж сражающийся с victim*	Нет	Вызывается каждый раунд битвы (каждые 3 секунды), после того как персонаж провел все свои атаки.
INIT разовое не отменяемые RANDOM TICK периодические не отменяемые	(RANDOM, TICK) Area: func(Area area) (INIT, RANDOM, TICK) Obj: func(Obj obj) - **Внимание: на момент создания носитель вещи или место респауна равно nil! TODO исправить, чтобы вещь всегда ссылалась или на комнату , чара или предмет-контейнер!** (INIT, RANDOM, TICK)Char: func(Char ch) *area, obj, ch- объект, обрабатывающий событие*	Нет	RANDOM - каждые 3 секунды TICK - каждую минуту (mud-tick). INIT - вызывается один раз при создании вещи/моба (старт сервера, репоп/респаун, вход в игру). *Примечание: для игроков сделано исключение*
RESET	Room: func(Room room)	Нет	Когда комната ресетится.
SOCIAL разовое отменяемое	Room: func(Room room, Char ch, строка cmd, Char victim, строка arg) Obj: func(Obj obj, Char ch, строка cmd, Char victim, строка arg) Char: func(Char mob, Char ch, строка cmd, Char victim, строка arg) *ch - персонаж, применяющий социал* *cmd - полное (всегда) английское название социала* *victim, персонаж-цель (или nil)*	Имя социала	Персонаж набрал существующий социал рядом или в объекте, у которого есть обработчик данного события.

	arg - строка с аргументами заклинания		
SPEECH разовое не отменяемое	Room: func(Room room, Char ch, строка speech) Obj: func(Obj obj, Char ch, строка speech) Char: func(Char mob, Char ch, строка speech) *room,obj, mob - объект, обрабатывающий событие* *ch - персонаж, которой произнёс фразу speech*	Регистронезависимый шаблон для совпадения с фразой	Персонаж произнес (путем исполнения команды say) фразу, совпадающую с шаблоном рядом/в объекте, где есть обработчик данного события.
BRIBE разовое отменяемое	Char: func(Char mob, Char ch, число amount) *mob - персонаж, обрабатывающий событие* *ch - персонаж, дающий деньги (взятку) mob* *amount - сумма денег, переведённая в серебряные монеты (*100). Если она меньше PARAM,персонаж их молча возьмет, а* ***событие не сработает****!	Минимальная сумма для срабатывания	Персонаж ch пытается дать mob деньги (взятку, оплату). Если нужна реакция на недостаточную сумму, не указывайте PARAM, а обрабатывайте внутри события.
DAMAGE разовое отменяемое	Char: func(Char ch, Char victim, число damage, строка skill) *ch - персонаж, наносящий повреждения (может быть nil! Если скриптом)* *victim, персонаж-жертва* *damage - количество очков здоровья* *skill - каким навыком/заклинанием наносится урон*	Имя скилла или заклинания	Вызывается при нанесении урона **у обоих персонажей** (при наличии обработчика(ов) события).
DEATH разовое отменяемое	Char: func(Char victim, Char ch) *victim, персонаж - умирающая жертва* *ch - персонаж, нанёсший фатальный удар (может быть nil! Если скриптом)Char: func(Char mob, Char ch, число amount)*	Нет	Жертва умирает hit points (здоровье) меньше 1, position = “dead”. При отмене установить hit points и position! Можно уничтожать жертву victim.destroy().
PEACE разовое не отменяемое	*Char: func(Char mob)*	Нет	Вызывается после окончании боя. Обязательная проверка на mob.position (может быть «dead»!)
HITPERCENT периодическое не отменяемое	Char: func(Char ch) *ch - персонаж, обрабатывающий событие*	Процент, ниже которого срабатывает	Вызывается каждый раунд битвы (каждые 3 секунды), если hit points персонажа ниже порога в PARAM.
GIVE Obj:разовое отменяемое Char: разовое не отменяемое	Obj: func(Obj obj, Char ch, Char victim) Char: func(Char victim, Char ch, Obj obj) *ch - персонаж, передающий предмет obj* *victim - персонаж-получатель предмета* *obj - вещь, которую передают (или у которого сработало событие)*	Имя или vnum предмета - **только для Char**	Персонаж ch передает персонажу victim предмет obj. Если сработал обработчик у Obj, то **можно отменить** передачу, для Char **нельзя**-предмет **уже** у получателя.
CRASH разовое отменяемое	OBJ: func(Obj obj, число damage) *obj - вещь, которая получит повреждения* *damage - повреждения в процентах*	Нет	Вызывается перед тем, как вещь получит повреждения (можно отменить).
DESTROY разовое не отменяемое	OBJ: func(Obj obj) **TODO никогда не использовался! Может удалить?** *obj - вещь, которая уничтожается физически*	Нет	Вещь будет уничтожена физически. Отменить нельзя!
FROMOBJ ONOBJ разовое отменяемое	(FROMOBJ) Obj: func(Obj obj, Char ch) (ONOBJ) Obj: func(Obj obj, Char ch, строка action) *obj - вещь, с которой слезают/встают/садятся/ложатся* *ch - персонаж, слезающий/встающий/садящийся/ложащийся на obj* *action - действие, которым пытаются забраться (stand/rest/sit/sleep)*	Имя действия (stand/rest/sit/sleep)	Персонаж ch слезает/влезает на вещь obj. Если спит на вещи, а потом садится на неё же, то FROMOBJ будет вызвано до ONOBJ.
GET PUT TAKEFROM разовое отменяемое	(GET)Obj: func(Obj obj, Char ch) (PUT, TAKEFROM) Obj: func(Obj obj, Char ch, Obj victobj) *obj - вещь, которую поднимают/в (из) который(ого) кладут/берут* *ch - персонаж, который поднимает, кладёт или достаёт объект* *victobj - вещь, которую передают (или у которой сработало событие)*	Имя или vnum предмета victobj (PUT, TAKEFROM)	Персонаж ch пытается подобрать (GET), положить/достать в/из вещи obj другой предмет victobj (PUT, TAKEFROM).
REMOVE WEAR разовое отменяемое	Obj: func(Obj obj, Char ch) *obj - вещь, которую пытаются снять/надеть* *ch - персонаж, который снимает/надевает предмет*	Нет	Персонаж ch пытается снять/надеть вещь obj. Событие возникает только если действие **возможно** осуществить!
TIMER периодический отменяемые	Obj: func(Obj obj)(INIT, RANDOM, TICK)Char: func(Char ch) *obj - предмет, у которого истёк таймер (таймер уменьшается на 1 в tick).*	Нет	Истёк внутренний таймер. Если отменить, то таймер обнулится, а вещь не исчезнет.
LOCK UNLOCK разовое отменяемое	Obj: func(Obj obj, Char ch, Obj victobj) Room: func(Room room, Char ch, строка dir) *obj,room — вещь или комната, которую отпирают/запирают* *ch - персонаж, который производит действие* *victobj - ключ. Может быть nil при открывании без ключа (мастер особняка/комнаты)* *dir -направление в комнате Room, в котором расположена дверь*	Имя или vnum предмета victobj, строка направления к открываемой двери dir	Вызывается у контейнера/комнаты при попытке запереть/отпереть контейнер obj/дверь в комнате room в направлении dir
USE разовое отменяемое	Obj: func(Obj obj, Char ch) *obj - объект, который пытаются использовать* *ch - персонаж, использующий объект*	Нет	Персонаж ch пытается использовать вещь obj. Срабатывание зависит от типа.

TODO MOBile добавить событие ENTER? не менее, обработчик вызывается до того как персонаж сделает look и, если его присутствие в данной комнате нежелательно, его можно переместить в другую комнату (функция Move класса CHAR) - он не заметит разницы. Проверить (Area Room) ENTER и возможность объединения с GREET (char), при этом GREET сделать отменяемым

TODO подумать, может сделать GIVE у Char отменяемым, как у OBJ - только нужно будет переделать все эвенты, потому что у Char вызывается уже после передачи, т. е. предмет уже у получателя Char. Будет удобнее с предметами, которые получатель не хочет брать - написал сообщение и статус возврата 1. А так нужно что-то мудрить со сбросом на

Примечание: событие Init при появлении игрока (в Мире, реморте и т.д.)

Так как игроки создаются "на лету" (в are-файлах нет описания мобов для игроков, тело моба создаётся как бы на лету, то для события INIT игрокам невозможно навесить обработчики заранее. Поэтому сделан принудительный вызов из кода (при заходе игрока в игру) функции "system_player_entergame_init" (расположенный в scripts.are), который позволяет внутри этой функции добавлять всем игрокам другие события командой Char.add_event(...). Например, можно использовать для инициализации клановых скиллов (if ch.clan == "Dbb" then вешать обработчик кланскилла)!

Тип объекта	Момент срабатывания события	Тип объекта	Момент срабатывания события
armor	надеть объект		
boat	войти с объектом в место, где нужна лодка	pill	съесть объект
clothing	надеть объект	portal	войти в объект
drink container	попить из объекта	protect	выпить жидкость
food	съесть объект		
fountain	попить из объекта	scroll	прочитать свиток
furniture	встать/лечь/сесть на объект	staff	взмахнуть посохом
gem	надеть объект	tattoo	надеть объект
graffiti	написать или стереть граффити	trash	надеть объект
jewelry	надеть объект	treasure	надеть объект
jukebox	включить музыку в объекте	wand	взмахнуть жезлом
light	надеть объект	warp stone	создать portal или nexus
		weapon tail_weapon	вооружиться объектом

Также событие возникает, если объектом любого типа пытаются отпереть или запереть двери/предмет и он подходит по vnum, как ключ. Для объектов money, key, roomkey и map это событие не вызывается (за исключением ключа к двери/предмету).

Правила написания скриптов.

Эти правила — условные, их несоблюдение не помешает скриптам выполнятся. Но всё-таки стоит придерживаться их, для своего и чужого удобства, лучшей читаемости кода, улучшения производительности, эффективного использования процессорного времени и минимизации времени для поиска ошибок.

Правило составления имён функций.

Так как функции разделены по ариям, то первым словом идет названия .are-файла (midgaard, limbo, newthalos и т.д.)

Смысловое название функций должно помогать понять, для чего эта функция предназначена без заглядывания в её код (.

Вспомогательные функции, не являющиеся обработчиками событий:
func_name = [префикс название are файла]_[смысловое название функции]

Примеры:
midgaard_

Функции, являющиеся обработчиками событий:
func_name = [префикс название are файла]_[смысловое название функции]_[тип события]_[тип N события]_[строка отбора PARAM, если есть]
Допускается совпадение смыслового названия функции и типа события:

Примеры:
midgaard_holidays_init - midgaar.are, событие init, holidays (англ. праздники) — расположенное в арии мидгаарда инициализация праздников.

Всё вышеприведённое позволяет понять где описана функция-обработчик, к кому относится или для чего предназначена, какими событиями вызывается.

Максимально используйте объект Lua – таблицу. Таблица является универсальным индексируемым средством хранения данных. В одной таблице можно хранить разнородные данные и индексировать их тоже по разнородным индексам - true/false, числовым, строковым, датам, ссылочным объектам (персонажам, предметам, ариям, комнатам) и даже по ссылкам на функции.

Выносите таблицы из функций в глобальные переменные — потому что таблица внутри функции создаётся и заполняется **каждый раз**, когда функция выполняется! Глобальная таблица создается один раз, при первоначальной компиляции скриптов. Правила именования таблиц похожи на правила именования функций-обработчиков: table_name = [префикс название are файла]_[предназначение таблицы]_table

Кэшируйте глобальные переменные внутри функций: local имя_таблицы = имя_таблицы. Казалось бы присваивание одного того же, но использование local даёт возможность Lua получать ссылку на таблицу за один шаг из сохранённой локальной переменной. Из глобальных переменных всё делается минимум за два шага — сначала получить глобальную таблицу ссылок на все таблицы/функции, потом из неё получить ссылку на таблицу. К сведению, функции Lua – это тоже ссылки из глобальной таблицы. То есть если у Вас в цикле выполняется за каждый проход обращение к методу объекта, к функции Lua – то имеет смысл закэшировать эти ссылки на функции/методы объектов — тогда их вызовы будет выполнятся быстрее. И чем больше итераций цикла, тем более это будет эффективней.
Пример:
midgaard_numbers_table = {“Привет!”, “Как дела?”, “Пока не родила...”}

```
function midgaard_echo(ch)
 -- кэшируем глобальную таблицу, кэшируем метод ch.act(), обьявляем локальную переменную i для цикла
     local mnt, ch_act, i = midgaard_numbers_table, ch.act
     for i = 1, #random_table do
        ch_act(“room”,   mnt[i])
     end
end
```

Не забывайте объявлять локальные переменные. Если переменная не объявлена через local переменная — то она считается глобальной и к ней доступ идёт через получение из глобальной таблицы! К тому же, другие функции могут изменить в ней значение, что приведет к не прогнозируемым сбоям.

Хорошенько изучите файлы stdlib.lua и script.are – в них содержаться функции, упрощающие и автоматизирующие многие действия. Не стоит изобретать свой велосипед с квадратными колёсами — он скорее всего уже есть.

Эффективней используйте события RANDOM и TICK. Не стоит делать обработчик, вызываемый каждую минуту (и уж тем более каждые 3 секунды) для того, чтобы он выполнил свою миссию всего несколько раз в день при десятках тысяч вызовов впустую. Как показывает практика, тоже самое можно легко сделать комбинируя другие события или используя следующий вариант со свойством timer класса Obj. Например, нужно сделать событие, которое создаёт моба-босса раз в сутки в случайное время. Вместо того, чтобы тупо сделать событие TICK и каждый раз ожидать у Mud.random_percent() == 1 (1% вероятность) разрешения на респаун, можно сделать минимум 2 варианта:

1) Повесить событие ENTER на арию. Как только игрок пытается войти в арию (а мобы предназначены для игроков) — создаём босса, если рандом даёт нам шанс (сама вероятность уже должна быть рассчитываемой по формуле - чтобы игрок-бот, догадавшийся о таком варианте респауна не делал на границе арии run nsnsn... Вероятность может зависить от общего счётчика посещений, от персонажа (допустим он ходит проверять, респаунился ли босс — тогда с каждым посещением ему можно увеличивать вероятность появления, учитывать интервал появления этого персонажа в арии. Сами счётчики можно хранить на персонаже). Это событие лучше совместить с какой-нибудь другой работой, связанной с появлением персонажей в арии.

2) Выбираем какой-нибудь объект в арии и прописываем ему свойство timer = 1 и событие TIMER. Как только ария будет загружена, на следующий тик счётчик закончится и вызовется событие TIMER. Допустим, если на объекте нет переменной obj.get_var('boss_timer') == nil, значит это первый вызов и босса респаунить не надо. В обработчике события вычисляем случайное время репопа босса и полученное число записываем в obj.timer = Mud.random_range(50, 610) – от 50 минут до 10 часов * 60 + 10 минут. Вызов заканчиваем obj.set_var('boss_timer', 1),

чтобы в следующий запуск босс был создан и return 1 – чтобы объект не был уничтожен. Поле timer будет уменьшаться в коде c++ каждую минуту и через указанное время опять последует вызов события TIMER. В котором создаём босса (boss_timer' не nil) и ставим следующий рандомный интервал до вызова проверки жив ли босс, и если не жив, опять создаём и так далее.

И в том и в другом варианте нет лишних вызовов кода Lua (второй вариант с этой точки зрения вне конкуренции). Если делать через TICK – это 24 часа * 60 минут = 1440 вызовов в сутки, куча процессорного времени, для подготовки, исполнения кода и обработки результатов из C++ в Lua и обратно. Некоторые кодеры ранее использовали событие RANDOM, вызывающееся каждые 3 секунды для таких вот целей, в 20 раз больше, чем TICK! Почувствуйте разницу! По-хорошему, событие RANDOM нужно назначать объектам для детализации события до интервалов в 3 секунды перед нужным результатом. После достижения результата - вызов события удалять. Или даже стоит сделать один вызов RANDOM на весь мир с диспетчером, который регистрирует запросы на вызовы от других функций в своей таблице (и вызывает, когда надо).

Зачем нужна проверка на валидность — valid() и проверка ссылок на nil.

Все ссылочные объекты имеют свой идентификатор-ссылку. При обращении к ней можно получать значения свойств и методов.
Но эти ссылочные объекты могут быть уничтожены (предметы - разрушены, мобы - убиты или уничтожены, игроки - выйти из игры, временные комнаты (dungeon, лабиринт, дуэльные зоны, школа для новичков и т.д.) - удалены. Возможно в будущем ещё и арии смогут выгружаться на лету). То есть переменная с запомненной ссылкой существует и не равна nil, но сама ссылка ведёт на уже несуществующий объект. Любое обращение к методам/свойствам такой ссылки вызывает ошибку. Поэтому в обработчиках, которые связаны с повреждениями, изменениями объектов, с функцией задержки delay или при создании ситуации с вложенным вызовом другого события — нужно проверять, не стал ли объект не валидным.

Пример 1:
```
   mob.damage(Mud.random_range(30, 250), "obj_hit", "poison", ch, "укус|ы|ов|ам|ы|ами|ах", "plural")
   mob.act("all resting", ch, "$C-Ti громко кричи$ушь|т от боли.")
```

Комментарий: При нанесении повреждений damage персонаж-жертва может умереть и последующая попытка обращения к методу act вызовет ошибку и останов работы обработчика события.

Пример 2:

event area_otello_say()
```
   mob.execute(Mud.random_range(30, 250), "obj_hit", "poison", ch, "укус|ы|ов|ам|ы|ами|ах", "plural")
    ch.act("all resting", ch, "$C-Ti громко кричи$ушь|т от боли.")
```

event area_desdemona_death

Выполняемые команды через execute могут вызвать другие обработчики, которые сделают объекты, используемые в текущем скрипте, не валидными (убить, уничтожить и т.д.)

Пример 3:
```
       i = 1
        while i < 7  then
            mob.one_hit(ch) -- выполнить 1 удар, от которого могут умереть и ch, и mob (от возвратки)
            i = i + 1
       end
```

Персонаж, от имени которого наносятся удары тоже может умереть, от возвращаемого ему повреждения (молитва Норхат возвращает от 10 до 15%, повреждений, умение Зеркало может отразить заклинание на того, кто его произнёс). Что сделать: добавить условия в while:

```
        while i < 7 and valid(ch) and valid(mob) then
```

Пример 4:
```
    ch.damage(1234, "obj_hit", "pierce", victim, "удар||а|у||ом|е хвостом {двиверны{х", "male")

    victim.wait = 6
    victim.daze = 6
```

Можно избегать лишнего кода, перенося строки кода, обращающиеся к объекту перед критичным местом (в вышеуказанном примере можно спокойно установить wait и daze перед вызовом damage - умрёт victim? В процедуре возрождения ему и так все поля очищают).

Классы LUA

Класс Mud

Свойства

У данного класса свойств нет.

Методы

Возврат	Название	Описание
нет	log(строка message)	Написать сообщение message в лог и на консоль сервера.
строка	money_to_text(число gold[, число silver[, булево convert_silver_to_gold = true]])	Числовые значения золота и, опционально, серебра, вернуть в виде строки «X золотых и Y серебрянных монет». При указании флага конвертирования, серебро свыше 99 монет конвертируется в поле gold.
число	mode()	Получить режим работы сервера в виде флагов: • 0x00000001 — скомпилирован в DEBUG-режиме • 0x00000002 — скомпилирован под Windows-платформу (иначе – Unix) • 0x00000004 — скомпилирован в DEMO-режиме • 0x00000008 — запущен в тестовом режиме (ключ -test в строке запуска)
число	random_range(число min, число max)	Получить случайное число из диапазона [min-max].
число	random_percent()	Получить случайное число (процент) в диапазоне [1-100].
число	reboot_time()	Получить время (в тиках), оставшееся до ребута.
число	table_size(таблица table)	Получить количество ВСЕХ элементов луа-таблицы (значение == nil не учитываются!).

Класс World

Свойства

Изм.	Тип	Название	Описание

Методы

Возврат	Название и параметры	Описание
строка	adj_name(число или символ case, [,булево char_not_obj = true[,булево remove_color = true]])	Получить имя моба (char_not_obj = true) или наименование предмета (false)с учетом падежа case в числовом [1-6], или односимвольном виде [iI1rR2dD3vV4tT5pP6] прямо из базы описания (объекты могут в Мире не существовать) . При создании имени моба можно получить чистое имя, без цветов (remove_color = true).
Affect	create_affect()	Создает список аффектов.
нет	destroy_affect(Affect affects)	Уничтожает список аффектов affects.
нет	echo(строка message)	Вывести строчку message всем в мире.
nil/Area	get_area(строка name)	По заданному имени возвращает арию или nil, если такой арии не существует. Если name не указан, вернуть первую арию из списка арий для последующего обхода.
nil/Char	get_char(строка name) get_char(число vnum)	Возвращает персонаж, имя или виртуальный номер которого совпадает с заданным. Если такой персонаж не найден, функция возвращает nil.
Area	get_limbo()	Возвращает комнату с vnum 2 из арии Limbo. Использовать как отстойник предметов.
Area	get_mansion()	Возвращает особняки.
nil/Obj	get_obj(число vnum)	Возвращает предмет по его vnum или nil, если такой предмет не существует.
строка,строка	get_race_info(строка race) get_race_info(Char ch)	Возвращает 2 значения — английское и русское название расы.
строка	get_random_obj_name	Возвращает имя случайного объекта типа type.
Room	get_random_room([Area area][, Char ch [, строка area_flags room_flags]])	Возвращает случайную комнату. Если указана area - то внутри неё, если указан персонаж ch, то исключаются арии с флагами godsonly hidden underconst noquest noquit testing и комнаты private imp_only gods_only newbies_only gray_wall prison labyrinth solytary depleted pet_shop. Если указана строка флагов, то пропустить такие арии/комнаты. Если за 10 попыток не удаётся найти подходящую комнату, то возвращается реколл (3001).
nil/Char	get_random_pc()	Возвращает случайного игрока в мире или nil, если не найлено. При выборе не учитываются те, у кого lostlink и Бессмертные.
Lua-таблица	get_rating()	Получить таблицу rating (рейтинг кланов).
nil/Room	get_room(число vnum)	Возвращает комнату по её vnum или nil, если такой комнаты не существует.
нет	helpers_print(строка message)	Вывести строчку message всем хелперам в мире.
nil/Char	players()	Получить первого pc-персонажа в мире. Возвращает nil, если никого в мире нет. (**TODO надо реализовать CHAR.next_in_world() для обхода всего списка.**)

Класс Affect

Свойства

Изм.	Тип	Название	Описание
да	число	duration	Длительность аффектов в пульсах (в 4 раза больше числа секунд). По умолчанию **ноль**. По истечению времени аффект удаляется. Если установить в **-1**, то аффект превратится в постоянный.
да	число [0-110]	level	Уровень аффектов. По умолчанию **ноль**.
да	строка	origin	Природа аффектов. Принимает значения из списка: spell, wand, staff, scroll, food, pill, potion, drink, secondary, room, hit, skill, raceskill, equipment, nature, quest, rune, script, bio, **unknown(по умолчанию)**.
да	строка	skill	Название скилла аффектов. По умолчанию — **не заполнен**.

Методы

Возврат	Название и параметры	Описание
нет	add_apply(строка name, число param)	Добавить числовой модификатор к параметру с именем name (может принимать одно из следующих значений: str, dex, int, wis, con, sex, level, mana, hp, move, ac, hr, dr, svs, heal bonus, size, position, spell, spell damage, resist/damage all/bash/и т. д.). **(TODO написать полный список!)**
нет	clear_apply()	Очистить список **всех** апплаев. Установленные свойства duration/level/origin **не очищаются**!

Пример использования класса Affect

```
local af = World.create_affect()  -- выделить память под аффект
      af.skill = "curse"          -- аффект вешает заклинание "проклятие"
      af.origin = "quest"         -- от origin зависит, например, будет ли сниматься cancellation
      af.duration = 16            -- длительность аффекта 4 секунды * 4
      af.level = ch.level         -- уровень аффектов будет равна уровню персонажа
      af.add_apply("level", -11)  -- аффект снижает уровень произносимых заклинаний на 11 уровней
      af.add_apply("move", -300)  -- аффект снижает количество очков движения на 300
      ch.apply_aff(af)            -- применить аффект(делает копию аффекта на персонаже)
      af.clear_apply()            -- удалить из аффекта всё, что выше туда добавили
                                  -- и можно использовать созданный аффект для других целей
      af.origin = "spell"         -- данное заклинание будет видно в списке команды affect
      af.skill = "bless"          -- и его можно будет отменить заклинанием cancellation
      ch.apply_aff(af)
World.destroy_affect(af)          -- если не удалить аффект, возникнет утечка памяти!
```

Класс Area

Свойства

Изм.	Тип	Название	Описание
нет	число	age	Возраст арии - количество тиков, прошедших с момента последнего обновления арии.
нет	Lua-таблица	date	Час hour в арии [0-23], день day [0-34], месяц month [0-16], год year [0-11], время gener [0-11], век cent [0-3]
нет	число,число	levels	Возвращает два значения — диапазон минимальный и максимальный рекомендованные уровни для посещающих арию игроков.
нет	строка, строка	name	Возвращает английское и русское название арии.
нет	nil/Area	next	Возвращает следующую арию из списка арий или nil, если эта ария была последней в списке.
нет	число	players_count	Количество игроков, находящихся в арии.
да	строка	resetmsg	Сообщение, показываемое при обновлении (reset) арии.
нет	nil/Room	rooms	Возвращает первую комнату в арии. Если в арии нет комнат, возвращает nil.
нет	число [0-4]	sun	Освещенность: темно(0), восход(1), светло(2), заход(3), неопределено(4) - для арий с флагом no_weather.
нет	число,число	vnums	Получить минимальный и максимальный виртуальные номера (vnum) мобов и комнат в арии.

Методы

Возврат	Название	Описание
булево	add_event(строка func_name, строка event_type[,int percent_chance[,строка param]])	Вешает арии функцию function_name (без префикса "lua:"!) на событие event_name. Необязательные параметры — шанс и параметр отбора.
булево	clone_rooms(таблица {Room/число vnum}, число vnum_delta)	Склонировать комнаты, с указанных в таблице в виде room или room_vnum, прибавив к vnum нужное смещение vnum_delta. Вернёт **Истина**, если удалось.
число	get_int_var(строка name)	Используйте этот метод, чтобы получить значение внешней числовой переменной name. Если переменная не найдена, то возвращается **ноль**.
Char	get_random_pc()	Получить случайного игрока в арии. Не учитываются те, у кого lostlink и **Бессмертные**.
nil/строка/ число/дата	get_var(строка name)	Метод, чтобы получить значение внешней переменной name типа строка, число, дата. Если переменная не найдена, возвращается nil.
булево	has_flag(строка name)	Возвращает true, если для арии установлен данный флаг и false, если нет.
булево	is_clone()	Возвращает true, если ария была склонирована от другой, иначе false.
нет	remove_event(строка func_name)	Удаляет все события, вызывающие функцию func_name (без префикса "lua:"!).
нет	remove_var(строка name)	Метод, чтобы удалить внешнюю переменную name.
нет	set_var(строка name, число value[, строка **ttl**]) set_var(строка name, строка string[, строка **ttl**])	Метод, чтобы создать и установить значение внешней переменной name типа строка, число, дата. Если **ttl** не пусто, то переменная сохраняется на указанное время. Для неограниченного хранения указывайте срок в 100 лет.TODO-ttl образ.

Класс Room

Свойства

Изм	Тип	Название	Описание
нет	Area	area	Определяет арию, к которой принадлежит комната.
да	булево	arena	Является ли комната ареной.
нет	число	chars	Количество персонажей (игроков и мобов) в комнате. (TODO объединить в 1 метод npcs,players,chars)
да	строка	clan	Клан, которому принадлежит эта комната. Только члены указанного клана смогут войти в комнату.?!!!
нет	nil/Obj	contents	Начало списка предметов, которые находятся в комнате. nil, если нет предметов на полу.
да	строка	descr	Описание комнаты.
нет	nil/Room	down	Получить комнату, в которую можно перейти, направляясь вниз.
нет	nil/Room	east	Получить комнату, в которую можно перейти, направляясь на восток.
нет	строка	graffiti	Текст сообщения (graffiti), написанного в комнате.
да	число	heal_rate	Скорость восстановления hp (в процентах) от нормы (по умолчанию 100).
нет	число	light	Степень освещенности комнаты, увеличивается на 1 для каждый персонажа со светильником.
да	число	mana_rate	Скорость восстановления mana (в процентах) от нормы (по умолчанию 100).
нет	строка	name	Название комнаты.
нет	nil/Room	next	Возвращает следующую комнату из списка комнат арии или nil, если эта комната была последней в списке.
нет	nil/Room	north	Получить комнату, в которую можно перейти, следуя на север.
нет	число	npcs	Количество мобов в комнате. (TODO объединить в 1 метод npcs,players,chars)
да	строка	owner	Владелец комнаты.Если у комнаты есть владелец, то только он сможет войти в неё (аналогично decanter).
нет	nil/Char	people	Начало списка персонажей, которые в данный момент находятся в комнате.
нет	число	players	Количество игроков в комнате. (TODO объединить в 1 метод npcs,players,chars)
да	строка	sector	Тип местности в комнате из списка inside, city, field, forest, hills, mountain, swim, noswim, unused, air, desert.
нет	nil/Room	south	Получить комнату, в которую можно перейти, направляясь на юг.
нет	nil/Room	up	Получить комнату, в которую можно перейти, направляясь вверх.
нет	число	vnum	Виртуальный номер (vnum) комнаты.
нет	nil/Room	west	Получить комнату, в которую можно перейти, направляясь на запад.

Методы

Возврат	Название	Описание
булево	add_event(строка name, строка event_type[,int percent_chance[,строка param]])	Вешает предмету функцию name (без префикса "lua:"!) на событие event_name с шанс percent_chance и отбором param.
нет	change_exit(string direction, string type, число vnum) change_exit(string direction, string type, Room room)	Создать выход из данной комнаты в направлении direction в комнату room, либо удалить выход, если room = nil. type может принимать два значения: twoway или oneway.
Char	create_mobile(число vnum)	Создать в комнате моба с указанным vnum.
Obj	create_object(число vnum)	Создать в комнате предмет с указанным vnum.
число	damage(число damage, string dtype, [Char victim, string noun, string sex])	Нанести урон dam типа dtype на чара victim или, если victim = nil, всем в комнате. Возвращает true, если урон был кому-то нанесён.
нет	echo(string message)	Вывести сообщение всем в комнате, аналог act("room"...).
nil/Room/ булево	exit(строка direction[, строка exit_flag[, булево value]])	Получить комнату в нужном направлении (nil - туда нет выхода), но если указан exit_flag (**door, closed, locked, pickproof, nopick, nopass, passproof, easy, hard, infuriating, noclose, nolock, oneside**) - вернуть значение, а если указано value, то и изменить.
nil/Char	get_char([число vnum][, число count]) get_char([string name][, число count])	Дать персонажа в комнате с именем name или vnum (Vnum **игроков** ноль!). Если не указаны, то первого чара в комнате. Если указан count, то это порядковый номер по списку (учитывая отбор).
строка	get_extra_descr(строка keywords)	Дать extra description предмета для указанного keywords.
число	get_int_var(строка name)	Получить значение переменной name. Если переменная не найдена или имеет не числовой тип, то возвращается **ноль**.
nil/Obj	get_obj(число vnum) get_obj(строка name)	Получить объект в комнате по имени name или vnum.
строка	get_path(число vnum[, Char ch]) get_path(Room room[, Char ch])	Получить путь в комнату room (только внутри текущей арии), если передан ch, проверяется флаги комнат на возможность ему пройти. Возможен возврат значений "#Buffer overflow" если путь слишком длинный или "#No way" если невозможно проложить путь.
nil/строка/ число/дата	get_var(строка name)	Получить значение внешней переменной name типа строка, число, дата. Если переменная не найдена, возвращается nil.
булево	has_flag(строка room_flag)	Установлен ли room_flag для комнаты или нет.
булево	is_magical_forest()	Растет ли в данной комнате магический лес.
nil/Char	random_char([строка target])	Получить рандомного персонажа, из комнаты. Если никого нет - nil. Можно конкретизировать ("char" [default] - все pc/npc, "player", "npc").
nil/строка	random_direction()	Получить рандомное направление движения из комнаты. Если из комнаты нет выходов, вернёт nil.
nil/Room	random_exit()	Получить рандомный выход из комнаты (nil - если выходов нет.)
нет	remove_event(строка func_name)	Удаляет все события, вызывающие функцию name (без префикса "lua:"!).
нет	remove_extra_descr(строка keywords)	Удалить extra description для указанного keywords.
нет	remove_var(строка name)	Метод, чтобы удалить внешнюю переменную name.
нет	set_extra_descr(строка keywords, строка value)	Установить extra description для ключевого слова keywords.
нет	set_flag(строка room_flag(, число set_unset))	set_unset <> 0 или не указан, установить room_flag, иначе сбросить.
нет	set_graffiti(string text, число countdown, число cost)	Установить текст сообщения (graffiti), длительностью countdown и стоимостью карандаша cost.
нет	set_var(строка name, число value[, строка **ttl**]) set_var(строка name, строка string[, строка **ttl**])	Метод, чтобы создать и установить значение внешней перемен-ной name типа строка, число, дата. Если **ttl** не пусто, то переменная сохраняется на указанное время. Для неограниченного хранения указывайте срок в 100 лет.**TODO-ttl образ.**

room_flag

Возвращает true, если в комнате установлен flag. flag может принимать одно из следующих значений: dark, no_mob, indoors, private, safe, solitary, pet_shop, no_recall, imp_only, gods_only, heroes_only, newbies_only, law, nowhere, free_kill, no_teleport, no_gate, pc_only, prison, laboratory, observatory, depleted, noquit, no_login, no_north, no_south, no_east, no_west, no_up, no_down, gray_wall, traps, labyrinth, return, clear_blood, dead, bio, no_transport, outdoors, light, non_existence, noquest, noconvoke, portals.

Класс Obj

Свойства

Изм	Тип	Название	Описание
нет	nil/Char	carried_by	Персонаж, у которого находится объект. Если nil, если предмет лежит на полу или внутри контейнера (см. in_obj).
да	число	charges	"Заряженность" объекта (луки, стаффы, ванды). Для других типов значение всегда ноль.
да	число [0-100]	condition	Степень поврежденности объекта. Примечание: Степень указывается в процентах от неповрежденного состояния.
нет	nil/Obj	contains	Начало списка предметов, хранящихся внутри объекта. Если пусто, возвращается nil.
да	число	cost	Стоимость объекта (в серебре).
да	число	count	Дать количество для предметов с флагом **count_heap, иначе всегда 1 штука**. Установить новое количество (для предметов с count_heap) или создать нужное количество (сам образец тоже считается, т.е. count = 10 это 1 исходный предмет + 9 новых (для них вызывается событие INIT). Если записать ноль, предмет уничтожается.
, да	строка	descr	Описание предмета. Примечание: Это описание будет выводиться при взгляде на объект.
да	строка	hittype	Тип удара (только для **Obj.type == weapon**).
нет	nil/Obj	in_obj	Предмет, внутри которого хранится объект. Если объект не находится внутри контейнера, тогда nil.
нет	nil/Room	in_room	Комната, в которой лежит предмет. Если предмет внутри контейнера или у какого-то персонажа, тогда nil.
да	число	level	Уровень объекта.
нет	строка	liq_game_name	Игровое название жидкости в объекте (с падежами), допустим, "шнапс||а|у||ом|е" **(TODO ни разу не использ-сь)**
нет	строка	liq_name	Неигровое название жидкости в объекте, к примеру, "schnapps"
да	строка	long_descr	Данное описание показывается игроку, когда он входит в комнату или набирает look. Если не указано, то объект не виден в комнате, но доступен, по названию (в формате арий это тэг long).
да	число	loot_id	Идентификатор добытчика. Если установить в ноль, то добытчиком становится тот, кто поднял вещь.
да	строка	material	Материал, из которого изготовлен объект.
нет	строка	name	Список названий объекта (ключевых слов - в формате арий это тэг keys).
нет	nil/Obj	next_content	Следующий предмет из списка предметов, хранящихся внутри объекта (используется при обходе списка в цикле).
нет	nil/Obj	on_obj	Предмет, на котором лежит объект. Если объект ни на чем не лежит, тогда nil.
да	строка	owner	Имя владельца вещи.
да	строка	short_descr	Краткое описание предмета (выводится в eq, inv, в формате арий это тэг name).
да	число	timer	Время существования объекта. Это время уменьшается с каждым тиком, и по достижении нуля вызывается событие TIMER - если оно вернёт 0, объект уничтожается. Если значение равно нулю, то у объекта нет таймера.
да	строка	type	Тип объекта ("armor", "container", "boat", "light" и т.п.) После смены типа объекта, значения value(0-4) обнуляются
нет	число	vnum	Виртуальный номер (vnum) объекта.
нет	строка	wear_flags	Место в экипировке, на которое можно надеть объект: none, light, lfinger, rfinger, neck1, neck2, body, head, legs, feet, hands, arms, shield, about, waist, lwrist, rwrist, wielded, hold, floating, secondary.
нет	строка	wear_loc	Место в экипировке, на которое надет объект: none, light, lfinger, rfinger, neck1, neck2, body, head, legs, feet, hands, arms, shield, about, waist, lwrist, rwrist, wielded, hold, floating, secondary.
да	число	weight	Вес объекта в кг, умноженный на 10 единиц (т. е. до 1 числа после запятой).

Методы

Возврат	Название	Описание
булево	apply_aff(Affect affects[, строка style])	Применить заполненный affects к вещи с видом добавления style, если указан: **cumulative** (old.(level*duration)+new.(level*duration))/new.level, **increment** - сложить все параметры, **harmful** - взять макс. значения, **useful** - заменить на новый.
нет	add_affect(строка apply, уровень level, число duration)	Добавить предмету аффект apply (bless, curse, invis, fireproof, vampiric, voice, razors, mana_spring, vorpal) уровня level и длительностью duration (-1 = вечный аффект).
нет	add_enchant_affect(строка apply, число duration, уровень level)	Добавить предмету аффект бонус (hitroll, damroll,ac) уровня level и длительностью duration (-1 = вечный аффект).
булево	add_event(строка func_name, строка event_type(,int percent_chance(,строка param)))	Вешает предмету функцию function_name (без префикса "lua:"!) на событие event_name. Необязательные параметры — шанс и параметр отбора.
строка	adj_name(число case) [1-6] adj_name(символ case) [iI1rR2dD3vV4tT5pP6]	Получить название предмета с учетом падежа case. Падеж может быть как в числовом, так и в односимвольном виде.
Obj	create_object(число vnum)	Создать в контейнере Obj предмет с указанным vnum и вернуть на него ссылку.
нет	destroy([число count_heap])	Уничтожить предмет (если предмет-куча, можно указать сколько в ней уничтожить).
нет	drop(число flag)	Бросить предмет из инвентори персонажа на пол (даже nodrop). Если flag указан и не ноль, объект бросается "тихо", без всяких сообщений. Срабатывает event DROP.
строка/число	get_based_data(строка data_field)	Получить исходное значение по-умолчанию для реквизита предмета, заданного ключевым словом (descr, long_descr, short_descr, level, weight)
строка	get_extra_descr(строка keywords)	Дать доп. описание (extra) предмета для указанного ключевого слова keywords.
число	get_int_var(строка name)	Используйте этот метод, чтобы получить значение внешней числовой переменной name. Если переменная не найдена, то возвращается **ноль**.

nil/строка/число/дата	get_var(строка name)	Метод, чтобы получить значение внешней переменной name типа строка, число, дата. Если переменная не найдена, возвращается nil.
булево	has_affect(строка apply)	Есть ли данный apply у объекта. TODO ссылку на расшифровку
булево	has_flag(строка item_flag)	Установлен ли **item_flag** для предмета или нет.
булево	has_weapon_flag(строка weapon_flag)	Установлен ли оружейный flag или нет. TODO объединить с has_flag
булево	is_weared()	Надета ли вещь.
нет	raise(число level) [0-110]	Поднять (или опустить) уровень объекта и все его параметры до указанного.
нет	remove(число silent)	Снять надетый предмет с персонажа (даже noremove). Если flag указан и не ноль, объект снимается "тихо", без всяких сообщений.
нет	remove_affect(строка apply)	Удалить apply (bless, curse, invis, fireproof, vampiric,voice,razors, mana_spring,vorpal).
нет	remove_event(строка func_name)	Удаляет все события, вызывающие функцию func_name (без префикса "lua:"!).
нет	remove_extra_descr(строка keywords)	Удалить доп. описание (extra) для указанного ключевого слова (keywords).
нет	remove_var(строка name)	Метод, чтобы удалить внешнюю переменную name.
нет	set_extra_descr(строка keywords, строка value)	Установить доп. описание (extra) для указанного ключевого слова keywords.
нет	set_flag(строка item_flag, число set_unset)	set_unset <> 0 или опущен, установить item_flag, иначе сбросить.
нет	set_value(число index [0-4], value)	Установить специфические параметры объекта(зависят от типа предмета).
нет	set_var(строка name, число value, (строка **ttl**)) set_var(строка name, строка string, (строка **ttl**))	Метод, чтобы создать и установить значение внешней перемен-ной name типа строка, число, дата. Если **ttl** не пусто, то переменная сохраняется на указанное время. Для неограниченного хранения указывайте срок в 100 лет.TODO-ttl образ.
нет	set_weapon_flag(?**??????????**)	Устанавливает оружейный флаг на оружии.(**TODO сделать сохраняемым**)
нет	to_char(Char ch)	Поместить предмет из любой точки Мира в inventory к конкретному персонажу (ch).
нет	to_obj(Obj obj)	Поместить предмет из любой точки Мира в конкретный объект (obj).
нет	to_room(Room room) to_room(число room_vnum)	Поместить предмет из любой точки Мира в конкретную комнату.
nil/число	value(число index [0-4])	Получить параметр по индексу (зависит от Obj.type). См. таблицу.
нет	wear() wear(строка location)	Надеть предмет. Объект должен находиться в inventory. Если указать (location) экипировки, то предмет **надевается на него, без всяких проверок**!

Параметры value(0-4) объектов, специфичные для каждого типа объекта (Obj.type):

Тип предмета	value(0)	value(1)	value(2)	value(3)	value(4)
armor	pierce	bash	slash	other	bulk[**0**-5] размер вещи
container	макс. вместимый вес[0, ..]	container_flags	container_key_vnum	max_one_weight[0, ..]	множитель[0, ..]
corpse	rank	vnum	corpse_id	corpse_flags	corpse_killer
drink	total[1, ..]	fill	liquid_type	liquid_flags	номер эссенции
food	fill[-200,200]	калорийность[-200,200]	food_id	food_flags	counter
fountain	не используется	не используется	liquid_type	liquid_flags	номер эссенции
furniture	max_users[0, ..]	max_weight[0, ..]	furniture_flags	hp[-10000, 10000]	mana[-10000, 10000]
key	key_from	key_to	не используется	не используется	не используется
jukebox	line	song	song_wait2	song_wait3	song_wait4
light	не используется	не используется	timer[-1, ..]	не используется	не используется
manual	percent[0,100]	spell	не используется	не используется	не используется
money	silver[0, ..]	gold[0, ..]	не используется	не используется	не используется
pccorpse	не используется	не используется	не используется	не используется	counter
portal	charges	exit_flags	portal_flags	target_room_vnum[-1, ..]	key_vnum[-1, ..]
potion/pill/scroll	spell_level[0-220]	spell1	spell2	spell3	spell4
staff/wand	spell_level[0-220]	max_charges[1, ..]	charges	spell1	spell2
seed	charges	не используется	не используется	не используется	не используется
trash	не используется	не используется	trash_id	не используется	counter
weapon/tail_weapon	weapon_class	weapon_ave[0, ..]	spell (для посохов)	hit_type	weapon_flags

item_flag Имеет ли объект данный флаг. flag может принимать значения:

Название	Описание	Название	Описание	Название	Описание
arcane	получен алхимией	rotdeath rot_death	разложится после смерти владельца	visdeath vis_death	часть тела (видна только после смерти)
auto	авто повышение статов	auto2	авто повышение статов2	norandom no_random	параметры фиксированы
antigood anti_good	не для добрых	antineutral anti_neutral	не для нейтралов	antievil anti_evil	не для злых
enchanted	улучшенно (энчант)	magic	создан или напитан магией	burnproof burn_proof fireproof fire_proof	защищён от огня и прочего.

bless	освящено	curse	проклято	evil	таит зло
countheap	вес определяется количеством в куче.	heap	одинаковые вещи собираются в 1 кучу (1 предмет с количеством)		
deleted	удалено, при загрузке персонажа изымается.	log	манипуляции с Obj пишутся в серверный лог.	[affects_checking]	[тикают_эффекты] системный флаг, не использовать
dark	тёмный.	glow	светится в темноте (виден без света в комнате)	invis	невидимый предмет
nolocate no_locate	не_обнаружимое	noflight no_flight	не_перелететь	nowhere no_where	не найти
hadtimer had_timer	исчезнет по окончанию таймера	meltdrop melt_drop	исчезнет если выбросить	meltremove melt_remove	исчезнет если снять
hidden	спрятано	hum	гудит, демаскируется в ловушке	trap	установлен в ловушке
nodrop no_drop	небросить	noget no_get	невзять	noput no_put	неположить
noremove no_remove	не снять	nouncurse no_uncurse	нерасколд	npc_only npconly	только для мобов
life_source	запасённая в предмете жизнь (отключено)	mana_source	запасённая в предмете мана	retain	засушенный труп или часть тела
blood_smell	добыто в драке	death_smell	добыто убийством	steal_smell	добыто воровством
no_resets noresets	не репопится при ресете арии	reboot_done	предмет уже был загружен 1 раз	reboot_once rebootonce	Репоп 1 раз в ребут
for_sell	для продажи (у продавца)	inventory unlimited	товар	sellextract sell_extract	при продаже изымается
nopurge no_purge	не уничтожимый	noquest no_quest noquests no_quests	не участвует в квестах	nosac no_sac	нельзя пожертвовать
nonmetal	большая часть не металл- ignore heat metal	nodisarm no_disarm tied	не обезоружить	nosteal no_steal	нельзя украсть
notsave not_save no_save nosave	не сохраняется при выходе из игры	stay_area stayarea	запрет выноса из арии репопа	unique	запрет надевать больше одной вещи
noloot no_loot	при смерти rpk - игрока не остаётся в трупе	pk	только для pk/rpk-игроков	red_pk	только для rpk-игроков
fly	Летающий предмет.	swim	плавает		

container_flags

corpse_flags

liquid_type

liquid_flags

food_flags

furniture_flags

exit_flags

portal_flags

weapon_class

weapon_flags

Класс Char

Свойства

Изм	Тип	Название	Описание
да	число	align	Порядочность персонажа. Примечание: Используйте данное свойство только если вам нужно знать абсолютное значение порядочности (-997, например, или 34).
нет	строка	all_classes	Список всех классов игрока. Примечание: Возвращает список классов, которыми уже играл игрок, разделенных пробелами (например "mage warrior"). Текущий класс персонажа в список не попадает. Список может быть пустым, если персонаж ни разу не проходил процедуру перерождения.
нет	число	carry_num_percent	Число предметов, который несет персонаж, в процентах по отношению к максимальному числу предметов, которое он может нести.
нет	число	carry_weight_percent	Текущий вес, который несет персонаж, в процентах по отношению к максимальному весу, который он может нести.
нет	Obj	carrying	Получить инвентарь персонажа.
да	строка	clan	Название клана, в котором находится персонаж. Если персонаж не в клане, значение clan равно пустой строке. Примечание: Присваивание данному свойству пустой строки делает персонажа неклановым. Мобы тоже могут быть в клане (например, клановые охранники).
да	число	clan_status	Должность персонажа в клане. Всего существует 11 статусов, от 0 до 10 (0 - минимальный, 10 - лидер клана).
да	строка	current_class	Текущий класс персонажа. Для мобов возвращается «mobile». Изменяется только у игроков.
да	число	constitution	Значение сложения персонажа.
да	строка	dam_type	Тип атаки моба (damage type).
да	число	damroll	Значение дополнительного damroll персонажа.
да	число	daze	Параметр "daze" персонажа. Примечание: Если параметр больше нуля, эффект от всех умений и заклинаний, применяемых персонажем, значительно уменьшается (с большой вероятностью не проходят заклинания, плохо срабатывают или не срабатывают умения). Значение параметра уменьшается на 4 каждую секунду.
да	строка	descr	Описание игрока.
да	число	dexterity	Значение ловкости персонажа.
да	число	drunk	Стадия алкогольного опьянения. Только у игроков.
нет	строка	ethos	Ethos игрока. Возвращаемая строка будет из следующего списка: "Unknown", "Lawful" , "Neutral", "Chaotic".
да	строка	facet	Грань персонажа (“healer”, “exorcist”, “inquisitor”) или none.
нет	Char	fighting	Персонаж, с которым мы сражаемся. Если это свойство неопределенное, персонаж в данный момент ни с кем не сражается.
да	число	full	Полнота желудка.
да	число	gold	Количество золотых монет (gold) у персонажа.
нет	Char	guarded_by	Кто охраняет данного игрока.
нет	Char	guarding	Возвращает персонажа, которого мы охраняем. Охрана заключается в том, что игрок следует за охраняемым персонажем и при попытке его атаковать игрок автоматически спасает персонажа и вступает в бой сам. Примечание: В данный момент охранять могут только игроки, команда охраны доступна только бессмертным. В будущем, возможно, будет сделано соответствующее умение для игроков.
да	число	hit	Текущее количество hit points персонажа.
да	число	hit_percent	Количество hit points персонажа в процентах от максимального.
да	число	hitroll	Значение дополнительного hitroll персонажа.
да	число	home	Виртуальный номер (vnum) "домашней" комнаты моба (комнаты, где он появляется).
да	число	hunger	Степень голода.
нет	Char	hunting	Цель, на которую в данный момент охотится персонаж. Примечание: Игроки не могут охотиться.
нет	число	id	Уникальный идентификатор персонажа. ID - это уникальный идентификатор-цифра (например, 10937162), которая уникально идентифицирует моба или игрока. Два моба с одинаковым vnum будут иметь разный ID в пределах ребута. Примечание: ID можно использовать для квестовых целей (когда игроку нужно убить именно этого монстра, а не любого монстра с таким vnum), поскольку его значение может быть сохранено во внешней переменной.
нет	число	incarnations	Количество перерождений игрока (для нереморта 1, один реморт - 2 и т.д.)
да	число	intellect	Значение интелекта персонажа.
да	число	last_death_time	Время, когда персонаж умирал последний раз. Это значение равно -1, если персонаж ни разу не умирал. Поскольку мобы умирают только раз, у них это значение всегда равно -1.
нет	число	last_divorce	Время последнего развода игрока. Примечание: Может быть -1, если игрок еще ни разу не разводился.
да	число	last_fight_time	Время, прошедшее с момента последнего удара, нанесенного персонажу или персонажем. Значение представляет собой объект класса TIME, который равен текущему времени, если персонаж в данный момент сражается, или времени окончания сражения. Через 10 секунд после окончания сражения сбрасывается в -1.
нет	число	last_level	Возвращает, в каком возрасте был получен последний уровень.

нет	Char	last_social	Кто последний "социалил" персонажу. Полезно для проверки - а направлен ли социал нужному персонажу. Пример проверки: if (ch.last_social and ch.last_social == target) - проверить направлен ли социал персонажа target персонажу ch.
нет	Char	leader	За кем следует (follow) данный персонаж.
да	число	level	Получить или установить уровень персонажа в пределах от 1 до 101 (игроки) или до 110(NPC).
да	строка	long_descr	Длинное описание (long description) моба показывается входящему в комнату или после команды look.
да	число	mana	Текущее количество mana персонажа.
да	число	mana_percent	Количество mana персонажа в процентах от максимального.
нет	строка	marryed	Имя супруга игрока. Если игрок неженат (незамужем), возвращает пустую строку.
нет	Char	master	Возвращает хозяина для **npc**, и очаровавшего игрока для **pc**.
да	число	max_hit	Максимальное значение hit points персонажа.
да	число	max_mana	Максимальное значение mana.
да	число	max_moves	Максимальное значение moves.
нет	число	money	Общее количество денег на всех банковских счетах у игрока.
нет	Char	mount_by	Всадник, или кто едет на персонаже.
нет	Char	mount_on	Лошадь, или на ком едет персонаж.
да	число	moves	Текущее количество moves персонажа.
да	число	moves_percent	Количество moves персонажа в процентах от максимального.
нет	строка	name	Имя персонажа. Персонажи-игроки имеют однословное имя (Ayaks, Cuguar, Sable), однако персонажи-мобы имеют несколько имен (минимум два) - так называемый "список имен". Реально в разных действиях показывается имя игрока и short description монстра. Чтобы не забивать программу кучей проверок, проще пользоваться членами act и adj_name класса CHAR.
нет	Char	next_in_room	Свойство используется в циклах, для получения следующего (за текущим) персонажа в данной комнате. Например, если нужно просмотреть всех персонажей в некоторой комнате.
нет	Char	next_player	Получить следующего pc-чара из списка.
да	Obj	on_obj	На каком объекте сидит/спит/отдыхает персонаж.
нет	число	pc_death	Сколько раз другие игроки убивали этого игрока.
нет	число	pc_killed	Количество других игроков, убитых этим игроком.
да	число	perm_damroll	Значение базового damroll персонажа.
да	число	perm_hitroll	Значение базового hitroll персонажа.
да	Char	pet	Домашнее животное (pet) персонажа. Если равно nil, значит домашнее животное отсутствует. Это поле может быть установлено только у игроков - мобы не имеют животных, хотя теоретически могут.
да	число	pkstatus	pk-статус персонажа: 0 - PK_none, 1 - PK_GREEN (не используется), 2 - PK_YELLOW, 3 - PK_RED.
да	число	played	Сколько часов наиграно.
да	строка	position	Положение персонажа - из списка "dead", "mortal", "incap", "stunned", "sleep", "rest", "sit", "fight", "stand".
да	число	practices	Количество практик у персонажа.
да	число	pumped	Состояние "адреналин" у игрока. В этом состоянии игрок не может входить в безопасные комнаты, не может пользоваться recall и хуже использует транспортные заклинания. Состояние устанавливается, когда игрок нападает на другого игрока, и сбрасывается само через некоторое время неактивности (порядка 2 минут).
да	число	qpoints	Количество квестовых очков у игрока.
да	строка	race	Раса персонажа.
нет	Room	room	Комната, в которой в данный момент находится персонаж. Это значение всегда определено.
да	строка	sex	Пол персонажа - значение из списка - "none", "male", "female" или, только у **npc**, "either".
да	строка	short_descr	Короткое описание (short description) моба, участвует во всех действиях моба - это его имя с падежами.
да	число	silver	Количество серебряных монет (silver) у персонажа.
нет	строка	size	Размер персонажа - значение из списка: "tiny", "small", "medium", "large", "huge", "giant".
да	число	strength	Значение силы персонажа.
да	строка	superclass	Суперкласс персонажа или none.
да	число	svs	Защита от заклинаний персонажа.
да	число	thirst	Степень жажды.
да	число	timer	Сколько секунд игрок неактивен (не вводил команд). Для мобов — время существования.
да	строка	title	Титул игрока.
да	число	trains	Количество тренировок у персонажа.
да	строка	true_sex	Реальный пол игрока, без учета модифицирующих аффектов sex.
нет	число	vnum	Виртуальный номер (vnum) моба.
да	число	wait	Параметр "wait" персонажа, уменьшается на 4 каждую секунду. Все введённые персонажем команды откладываются до того момента, пока этот параметр не станет равным нулю.
да	число	wimpy	Значение "трусости" игрока, при котором он сбегает из боя.
да	число	wisdom	Значение мудрости персонажа.
нет	Area	zone	Зона, в которой находится "домашняя" комната моба.

Методы

Возврат	Название	Описание
нет	act(строка flags, строка msg) act(строка flags, Char victim, строка msg) act(строка flags, Obj obj, строка msg) act(строка flags,Char victim,Obj obj,строка msg) act(строка flags,Obj obj,Char victim,строка msg) act(строка flags, Obj obj1, Obj obj2,строка msg)	Используйте этот метод, чтобы вывести специфическую игровую информацию о персонажах или объектах и действиях с ними на основе текстового шаблона msg. Подробности см. ниже.
булево	add_event(строка func_name, строка event_type[,int percent_chance[,строка param]])	Вешает функцию function_name (без префикса "lua:"!) на событие event_name, с шансом (умолч.-100%) и параметр отбора. Если успешно, вернёт true.
булево	action(таблица lua_table, Char victim[,Obj obj] [,любой тип paramN])	Выполнить последовательность действий, указанной в луа-таблице. Вернёт true, если скрипт выполнился до конца, false - был прерван. См. подробности.
строка	adj_name(число или символ case, [,булево remove_color = true])	Получить имя моба/игрока с учетом падежа case. Падеж может быть как в числовом 1-6], так и в односимвольном виде [il1rR2dD3vV4tT5pP6]. При создании имени моба можно получить чистое имя, без цветов (remove_color = true).
булево	apply_aff(Affect affects[, строка style])	Применить заполненный affects к персонажу с видом добавления style, если указан: **cumulative** (old.(level*duration)+new.(level*duration))/new.level, **increment** - сложить все параметры, **harmful** - взять максимальные значения, **useful** - заменить на новый.
нет	apply_affect(строка effect, число level, число damage, строка origin)	Применить к персонажу effect ('acid', 'cold', 'fire', 'poison', 'shock') с выдачей соотв. сообщений (ослеплён дымом, обморожен, отравлен и т.д.).
нет	apply_sleep(число duration, Char victim)	Сколдовать sleep на персонажа victim левелом duration. (TODO нужно ли вообще?!!)
булево	attack(Char victim)	Используйте этот метод, чтобы заставить персонаж напасть на другого персонажа (victim). Если это возможно возвращается true. (TODO можно напасть в Safe. Проверить - зачем).
булево	being_crushed()	Возвращает, крашат ли (скилл crush) данного персонажа.
булево	can_see_char(Char victim) can_see_obj(Obj obj) can_see_room(Room room)	Проверить видит ли персонаж жертву (victim), объект (obj) или комнату (room). (TODO - сделать единую функцию can_see, которая определив тип параметра, сравнивает!!!)
булево	cast(строка spell, число level, строка target) cast(строка spell, число level, Char victim) cast(строка spell, число level, Obj target)	Применить воздействие заклинания, кастуемое персонажем, на цель (target). Методы не проверяют, доступно ли заклинание персонажу, достаточно ли у него маны и т. п. Возвращает true, если заклинание удалось скастовать.
число	check_resist(строка dtype)	Возвращает числовую величину сопротивляемости к типу урона dtype.
Obj	create_object(число vnum)	Создать в инвентаре персонажа объект vnum с базовым уровнем (level).
булево	damage(число dam, [строка att_type, строка dtype[, Char victim,[строка noun,[строка sex]]]])	Нанести урон dam чару victim/victim = Char), если victim = nil, тогда всем в комнате. att_type может принимать значения char_hit, obj_hit, любого навыка/заклинания или значение из таблицы Table_attack_types_and_damage_types. Возвращает, удалось ли нанести урон хотя бы 1 персонажу.
нет	destroy()	Используйте этот метод, чтобы уничтожить персонажа. Примечание: Персонаж-игрок выйдет из игры и отключится от сервера, то персонаж-моб удалится. Этот метод не работает для Бессмертных.
нет	dismount()	Заставить спешиться персонажа или тому, на ком едут, ссадить всадника.
нет	execute(строка command)	Используйте этот метод для выполнения персонажем некоторой команды (command). Социалы и некоторые запрещённые в скриптах команды выполняться не будут.
строка	get_based_data(строка data_field)	Получить исходное значение по-умолчанию для реквизита моба, заданного ключевым словом (descr, long_descr, short_descr)
число	get_int_var(строка name)	Вернуть значение числовой переменной, если есть и тип число, иначе - 0.
Obj	get_plushka()	Получить случайный ценный объект для данного персонажа (см. help plushka).
nil/Char	get_random_fight(param)	Используйте этот метод, чтобы случайным образом получить одного из противников, с которым сражается персонаж.
nil/Char/ Obj	get_target(строка arg[,булево mobile = true])	По аргументу (см. help argument) вернуть игрока в комнате или предмет(у игрока или в комнате). Если mobile не указан или true, вернуть игрока в комнате, иначе предмет.
строка	get_target_name(Char target)	Возвращает имя цели (в том числе оборотня в форме), по которому можно применить команду к цели. Цель должна находится в одной комнате с персонажем.
nil/врем я/строка /число	get_var(строка name)	Вернуть значение переменной, если она была установлена. Если не была установлена, или в ней было сохранено nil - возвращает nil.
нет	give_exp(число count)	Дает опыт персонажу или, если count < 0, отнимает.
булево	has_flag(строка flag, строка value)	Установлено ли во флаге flag (**npc** = **act, off**, **npc+pc** = **imm, res, vuln, form, parts**, **pc** = **plr, comm, config**) значение value.
nil/Obj	has_item(строка type)	Есть ли объект типа type в инвентаре у персонажа. (НИ РАЗУ НЕ ИСПОЛЬЗОВАЛСЯ)
nil/Obj	has_obj(число vnum [, число count]) has_obj(строка name [, число count])	Есть ли в инвентаре достаточное число объектов, возвращает последний(если указан count - то вернёт объект по номеру count, даже если их больше).
булево	has_shell()	Использует ли персонаж панцирь (скилл shell).
нет	hunt(Char victim)	Задать **npc** цель для поиска в мире (используя gate/portal) и нападения на victim.
булево	is_affected(строка affect)	Используйте этот метод, чтобы проверить находится ли персонаж под воздействием заклинания или умения (affect).
булево	is_awaken()	Возвращает, находится ли персонаж в сознании (Char.position = rest|sit|fight|stand).

булево	is_charmer_here()	Возвращает истину, если **pc**, находится под действием "charm", видит своего чармера и находится с ним в одной комнате, иначе ложь.
булево	is_crushing()	Возвращает, крашит ли (скилл crush) персонаж кого-то.
булево	is_evil() is_good() is_neutral()	Является ли персонаж злым/добрым/нейтралом.
булево	is_in_twit(Char victim)	Находится ли персонаж в твитлисте у victim.
булево	is_killed()	Убит ли персонаж. Стоит проверять после вызова методов cast и damage!!!
булево	is_morphed()	В форме ли персонаж (lycanthrope); ненулевое значение, если - да, иначе - 0.
булево	is_newbie()	Является ли персонаж новичком (newbie), ненулевое значение, если - да, иначе - 0.
булево	is_immortal() is_npc() is_pc()	Является ли персонаж **Бессмертным (immortal)**, мобом (npc), игроком (pc). Использовать, если свойство описано только для мобов или игроков.
булево	is_shopkeeper()	Является ли моб продавцом (shopkeeper).
булево	is_sleeping()	Спит ли моб (Char.position == “sleep”)
булево	kill(Char victim)	Убить персонажа, при этом не выдаётся никаких сообщений, поэтому нужно самим сообщить "$c-Ti убит$a!". Если **npc**, то после смерти Char будет уничтожен.
нет	move(Room room[, bool AutoLook]) move(число room_vnum[,bool AutoLook])	Переместить персонажа в указанную комнату. Если AutoLook = true (по умолч. false), тогда, после перемещения, автоматически выполняется Char.execute(“look auto”)
нет	move_with_group(Room room[, bool AutoLook]) move_with_group(число room_vnum[, bool AutoLook])	Аналог Char.move - для персонажа и членов его группы, находящихся рядом с ним в одной и той же комнате.
нет	one_hit(Char victim[, [строка dtype], строка weapon_loc])	Заставить персонаж нанести один удар (провести одну атаку) по victim. Dtype может быть неопределен. Weapon_loc только «wielded/second». См. также методы damage().
нет	raise(число level)	Поднять/опустить **npc** и все его характеристики с текущего на +/- level уровней.
булево	remove_affect(строка affect)	Вернёт true, если удалось снять affect с персонажа и false, если его не было вообще). Т.к. сообщения не выводятся, выведите что-то вроде "Твоя кожа стала нормальной."
булево	remove_event(строка name)	Удаляет обработчик событий name. Вернёт true, если удалось и false, если не нашли.
нет	remove_var(строка name)	Удалить переменную name.
нет	restore_affects()	Восстановить родные аффекты персонажа.
нет	set_flag(строка flag, строка value, (число fset))	Установить/снять во флаге flag значение value.
нет	set_var(строка name,**value**[, строка ttl])	Установить переменную name в значение **value** (может быть следующих типов: число,строка,дата,Obj,Char,Room,Area). Если указан ttl, то переменная сохраняется при выходе из игры/при reboot для **pc** и **npc**.
число	skill_percent(строка skill[, число percent])	Получить процент умения skill, а если указан percent, то и изменить значение.
булево	social(строка social_name[, Char victim])	Выполнить социал. Если есть victim, то social victim. Если victim=персонаж,то social self. Вернёт true, если удалось выполнить, иначе false (victim спит/не найден и т.д.).
нет	stop_fight()	Заставить персонажа прекратить бой.
nil/Obj	wears_obj(число vnum[, число count]) wears_obj(строка name[, число count])	Надето ли достаточное число объектов (если count не указан, то проверяет на один предмет), возвращает последний объект с индексом count, иначе nil.
булево	wear_set(таблица lua_table)	Сет надет полностью? В таблице указываются числовые vnum предметов сета. Если нужно учитывать парные предметы (на шею, на руки) указать vnum дважды.

Метод DAMAGE нанесения повреждений

// без вывода сообщения, персонаж мочит сам-себя

INT CHAR.Damage (INT dam)

// для вывода стандартных сообщений, либо сам-себя, либо сам-врага

INT CHAR.Damage (INT dam, STRING att_type, STRING dam_type)

INT CHAR.Damage (INT dam, STRING att_type, STRING dam_type, CHAR vch)

// для вывода нестандартных сообщений (для случая сам-себя ch вместо vch)

INT CHAR.Damage (INT dam, STRING att_type, STRING dam_type, CHAR vch, STRING msg, STRING sex)

Используйте этот метод, чтобы нанести персонажу повреждения произвольного характера (в отличии от OneHit, который просто делает удар первым или вторым оружием). Примечание: После применения метода из-за возможной смерти персонажа, используйте функцию valid(персонаж), чтобы определить, является ли объект всё ещё определенным.

В методах, где не указан vch (или ch = = vch) считается, что персонаж наносит повреждения сам себе. Учитываются все иммунитеты, сопротивляемости, все проверки: на charm, на флаг, на запоминание, на начало боя, на safe, на смерть и т.п.

Параметр att_type служит для указания типа атаки, который определяет выводимое сообщение. Может принимать значения любого стандартного типа атаки, заклинания или навыка. Также, может принимать специальные значения char_hit, obj_hit (см. ниже).

Стандартные типы атаки и их сообщения:

att_type	Сообщение	dam_type
none	удар||а|у||ом|е	none
slice	скользящ|ий|его|ему|ий|им|ем удар||а|у||ом|е	slash
stab	выпад||а|у||ом|е	pierce
slash	рассекающ|ий|его|ему|ий|им|ем удар||а|у||ом|е	slash
whip	хлестк|ий|его|ему|ий|им|ем удар||а|у||ом|е	slash
claw	удар||а|у||ом|е когтями	slash
blast	разрушительн|ый|ого|ому|ый|ым|ом удар||а|у||ом|е	bash
pound	тяжел|ый|ого|ому|ый|ым|ом удар||а|у||ом|е	bash
crush	дробящ|ий|его|ему|ий|им|ем удар||а|у||ом|е	bash
grep	удушающ|ий|его|ему|ий|им|ем захват||а|у||ом|е	slash
bite	укус||а|у||ом|е	pierce
pierce	колющ|ий|его|ему|ий|им|ем удар||а|у||ом|е	pierce
suction	всасыван|ие|ия|ию|ие|ием|ии	bash
beating	молотящ|ий|его|ему|ий|им|ем удар||а|у||ом|е	bash
digestion	едк|ий|ого|ому|ий|им|ом плев|ок|ка|ку|ок|ком|ке	acid
charge	прям|ой|ого|ому|ой|ым|ом удар||а|у||ом|е	bash
slap	шлеп|ок|ка|ку|ок|ком|ке	bash
punch	удар||а|у||ом|е кулаком	bash
wrath	гнев||а|у||ом|е	magic
magic	маги|я|и|и|ю|ей|и	magic
divine	свят|ая|ой|ой|ую|ой|ой энерги|я|и|и|ю|ей|и	holy
cleave	раскалывающ|ий|его|ему|ий|им|ем удар||а|у||ом|е	slash
scratch	царапающ|ий|его|ему|ий|им|ем удар||а|у||ом|е	pierce
peck	укол||а|у||ом|е клювом	pierce
peckb	удар||а|у||ом|е клювом	bash
chop	рубящ|ий|его|ему|ий|им|ем удар||а|у||ом|е	slash
sting	жалящ|ий|его|ему|ий|им|ем удар||а|у||ом|е	pierce
smash	удар||а|у||ом|е изо всех сил	bash
shbite	шокирующ|ий|его|ему|ий|им|ем укус||а|у||ом|е	lightning
flbite	опаляющ|ий|его|ему|ий|им|ем укус||а|у||ом|е	fire
frbite	леденящ|ий|его|ему|ий|им|ем укус||а|у||ом|е	cold
acbite	окисляющ|ий|его|ему|ий|им|ем укус||а|у||ом|е	acid
chomp	мощн|ый|ого|ому|ый|ым|ом удар||а|у||ом|е	pierce
drain	высасывани|е|я|ю|е|ем|и жизненной силы	negative
thrust	умел|ый|ого|ому|ый|ым|ом выпад||а|у||ом|е	pierce
slime	плев|ок|ка|ку|ок|ком|ке слизью	acid
shock	электрическ|ий|ого|ому|ий|им|ом удар||а|у||ом|е	lightning
thwack	удар||а|у||ом|е с размаху	bash

flame	плам|я|ени|ени|я|енем|ени	fire
chill	мороз||а|у||ом|е	cold
bash	туп|ой|ого|ому|ой|ым|ом удар||а|у||ом|е	bash
hit	удар||а|у||ом|е	bash

Типы атаки заклинаний или навыков и их сообщения:

Атака	**Сообщение**
acid blast	поток||а|у||ом|е кислоты
burning hands	пылающ|ие|их|им|ие|ими|их рук|и||ам|и|ами|ах
call lightning	молни|я|и|и|ю|ей|и
cause critical	cause critical
cause light	cause light
cause serious	cause serious
harm	harm
chain lightning	молни|я|и|и|ю|ей|и
chill touch	ледян|ое|ого|ому|ое|ым|ом прикосновени|е|я|ю|е|ем|и
colour spray	радужн|ый|ого|ому|ый|ым|ом луч||а|у||ом|е
curse	прокляти|е|я|ю|е|ем|и
demonfire	демон|ы|ов|ам|ов|ами|ах
earthquake	землетрясени|е|я|ю|е|ем|и
energy drain	energy drain
fireball	огненн|ый|ого|ому|ый|ым|ом шар||а|у||ом|е
flamestrike	огненн|ый|ого|ому|ый|ым|ом шквал||а|у||ом|е
lightning bolt	молни|я|и|и|ю|ей|и
magic missile	магическ|ий|ого|ому|ий|им|ом заряд||а|у||ом|е
ray of truth	свет||а|у||ом|е истины
shocking grasp	шокирующ|ее|его|ему|ее|им|ем прикосновени|е|я|ю|е|ем|и
acid breath	едк|ое|ого|ому|ое|им|ом дыхани|е|я|ю|е|ем|и
gas breath	ядовит|ое|ого|ому|ое|им|ом дыхани|е|я|ю|е|ем|и
fire breath	огненн|ое|ого|ому|ое|им|ом дыхани|е|я|ю|е|ем|и
frost breath	леденящ|ее|его|ему|ее|им|ем дыхани|е|я|ю|е|ем|и
lightning breath	шаров|ая|ой|ой|ую|ой|ой молни|я|и|и|ю|ей|и
backstab	удар||а|у||ом|е
bash	сбивающ|ий|его|ему|ий|им|ем удар||а|у||ом|е
dirt kicking	брос|ок|ка|ку|ок|ком|ке грязью
kick	удар||а|у||ом|е ногой
weapon voice	оружи|е|я|ю|е|ем|и
charge	удар||а|у||ом|е с разгону
shield hit	удар||а|у||ом|е щитом
life spring	поток||а|у||ом|е энергии
violent sun	неистов|ое|ого|ому|ое|ым|ом солнц|е|а|у|е|ем|е
burning grounds	горящ|ая|ей|ей|ую|ей|ей земл|я|и|е|ю|ей|е
ice spears	ледяны|е|ми|м|е|ми|х коп|ья|ий|ьям|ья|ьями|ьях
freezing	замораживани|е|я|ю|е|ем|и
petrifaction	окаменени|е|я|ю|е|ем|и
tornado	смерч||а|у||ем|е
bow	стрел|а|ы|е|у|ой|е
weapon trap	оружи|е,|я,|ю,|е,|ем,|и, спрятанн|ое|ым|ому|ое|ым|ом в ловушке,
blackjack	оглушающ|ий|его|ему|ий|им|ем удар||а|у||ом|е
vorpal	точн|ый|ого|ому|ый|ым|ом удар||а|у|ом|е
exorcism	экзорцизм||а|у||ом|е
necro contagion	лихорадк|а|и|е|у|ой|е
upas	яд||а|у||ом|е
mana drain	прокляти|я|й|ям|я|ями|ях
general purpose	пул|я|и|е|ю|ей|е
high explosive	взрывчатк|а|и|е|у|ой|е

Параметр dam_type всегда должен быть задан, для определения типа повреждений, чтобы можно было проверить иммунитет или сопротивляемость. Может принимать только следующие значения:

dam_type	**Тип повреждения**
none	Тип неизвестен. От этого типа повреждения нет защиты!

bash	Удар
pierce	Укол
slash	Разрез
fire	Огонь
cold	Холод
lightning	Молния
acid	Кислота
sound	Звук
air	Воздух
earth	Земля
max	????
poison	Яд
disease	Болезнь
negative	
holy	
energy	Энергия
mental	Ментальный
water	Вода
light	Свет
other	
harm	Боль
charm	Очарование

Нестандартный (произвольный) тип сообщения можно вывести в двух случаях, когда att_type принимает значения char_hit или obj_hit. Только тогда будут использоваться параметры msg и sex.

char_hit - Некто ch наносит нестандартное повреждение vch (возможно ch == victim). Название повреждения с падежами надо прописать в параметр msg, а род - в sex.

obj_hit - Тоже самое, что char_hit, но без вывода источника повреждения. То есть либо источник не надо показывать, либо он анонимен, используется для повреждений от объектов или комнат, например.

msg - Название повреждения с падежами, по аналогии с таблицами описаными выше.

sex - Определяет подстановку Твой/Твоя/Твои для описаного типа повреждения.

Метод ACT вывода сообщений

VOID Char.act (STRING target, ARG1, ARG2, STRING act_string)

ARG1, ARG2 могут являться объектами типа CHAR, OBJ или быть неопределены в следующих сочетаниях:

// не заданы аргументы

VOID CHAR.Act (STRING target, STRING act_string)

// задан только ARG1

VOID CHAR.Act (STRING target, CHAR vch, STRING act_string)

VOID CHAR.Act (STRING target, OBJ obj, STRING act_string)

// заданы ARG1 и ARG2

VOID CHAR.Act (STRING target, CHAR vch, OBJ obj, STRING act_string)

VOID CHAR.Act (STRING target, OBJ obj, CHAR vch, STRING act_string)

VOID CHAR.Act (STRING target, OBJ obj, OBJ obj2, STRING act_string)

Используйте это метод, чтобы вывести специфическую игровую информацию о персонажах или объектах и действиях с ними.

Параметр target, определяет аудиторию получателей сообщения и имеет следующий синтаксис:

"<цель>[<минимальная позиция цели>]"

Цель сообщения должна быть указана всегда, а вот минимальная позиция может и отсутствовать. Минимальная позиция значит, что "начиная с данной позиции и всем имеющим больший номер". В данном случае следует исходить из логики скрипта.

Доступные цели сообщения:

- char - только инициатор метода ch
- victim - только персонаж-цель vch
- notvict - всем в комнате, кроме инициатора метода ch и персонажа-цели vch
- room - всем в комнате, кроме инициатора метода ch
- all - всем в комнате

Позиции, начиная с которых будет видно сообщение:

1. dead
2. mortal
3. incap
4. stunned
5. sleeping
6. resting
7. sitting
8. fighting
9. standing

Параметр act_string - строка вывода, может (должна) содержать специальные символы, начинающиеся с $, которые во время вывода будут заменены своими значениями. Таблица специальных символов и их значение приводится ниже.

Примечание: Lua, при работе со строками преобразует некоторые комбинации, начинающиеся с обратного слэша (magic symbols) в совершенно другие символы. К примеру, \" это для двойных кавычек, \' это апостроф, \\ это обратный слэш, \[левая квадратная скобка, \] правая квадратная скобка, \a звуковой сигнал, \b - действие, забить предыдущий символ (backspace), \e это escape (27 = 0x1Bh), \f formfeed, \n новая строка (10 = 0x0Ah), \t горизонтальная табуляция (смещение) - для муда, это 4 пробела, \r перевод каретки (13 = 0x0Dh), \v вертикальное табуляция (смещение). При рисовании ASCII-риунков не забывать, что одиночный \ нужно преобразовывать в \\, чтобы вышеприведённые символы рисовались как было задумано.

Падежи идут в следующем порядке:

1. именительный (тут стоит кто/что), синоним i,I
2. родительный (тут нет кого/чего), синоним r,R
3. дательный (дать это кому/чему), синоним d,D
4. винительный (ты видишь кого/что), синоним v,V
5. творительный (ты восхищен кем/чем), синоним t,T
6. предложный (ты говоришь о ком/чем), синоним p,P

Условные обозначения для параметров act_string:

- ch - персонаж, вызвавший метод - всегда доступен
- vch - персонаж-цель, по отношению к которому строится сообщение
- obj - первый объект задействованный в сообщении

- obj2 - второй объект задействованный в сообщении
- arg1, arg2 - если параметр применим для любого типа

Простые параметры act_string:

- $$ - знак '$'
- $c<П>, $C<П> - имя ch, vch в указаном падеже, в зависимости от условий м.б. short desc моба, "некто", "Бессмертный" и т.п.
- $o<П>, $O<П> - подставляет название obj, obj2 в нужном падеже, в зависимости от условия м.б. "нечто"
- $e, $E - местоимение оно/он/она/они, в зависимости от пола ch, vch
- $m, $M - местоимение ему/ему/ей/им, в зависимости от пола ch, vch
- $s, $S - местоимение его/его/её/их, в зависимости от пола ch, vch
- $h, $H - местоимение it/him/her/them, в зависимости от пола ch, vch
- $a, $A - автоматическое окончание глагола о//а/и, в зависимости от пола ch,vch
- $b, $B - автоматическое окончание глагола, для obj, obj2
- $g<ср_род|муж|жен|множ> - выбор слова из списка, в зависимости от пола ch
- $G<ср_род|муж|жен|множ> - выбор слова из списка, в зависимости от пола vch
- $f<ср_род|муж|жен|множ> - выбор из списка для obj
- $F<ср_род|муж|жен|множ> - выбор из списка для obj2
- $y - выбор, что показывать ch или всем остальным в комнате
- $Y - выбор, что показывать vch или всем остальным в комнате
- $D - первое слово из arg2
- $t, $T - процитировать arg1, arg2
- $z<П>, $Z<П> - процитировать arg1, arg2 в указаном падеже

Составные параметры act_string имею следующий вид:

$<C|c>-<t|T|s|S><П>

- c,C точно так же указывают на ch, vch в зависимости от необходимости;
- t,s - тебе/себе если видит хозяин имени, или имя для всех остальных зрителей (все с учетом падежа);
- T,S - тоже самое, только Тебе/Себе с большой буквы;
- <П> - номер падежа или буква-синоним.

Во многих случаях это позволяет ограничиваться использованием одного или максимум двух вызовов метода Act, вместо того, чтобы перебирать все аудитории целей. Используйте составные параметры крайне аккуратно.

Например, при вызове метода персонажем Leo с целью Хассан (то есть один из аргументов класса CHAR), по всем возможными сочетаниям по падежам и получателям, мы увидим следующие результаты возвращаемые составным параметром:

Падеж	**$C-T**	**$C-t**	**$c-T**	**$c-t**	**$C-S**	**$C-s**	**$c-S**	**$c-s**
Инициатор увидит								
Именительный	*Хассан*	*Хассан*	*Ты*	*ты*	*Хассан*	*Хассан*	*Ты*	*ты*
Родительный	*Хассана*	*Хассана*	*Тебя*	*тебя*	*Хассана*	*Хассана*	*Себя*	*себя*
Дательный	*Хассану*	*Хассану*	*Тебе*	*тебе*	*Хассану*	*Хассану*	*Себе*	*себе*
Винительный	*Хассана*	*Хассана*	*Тебя*	*тебя*	*Хассана*	*Хассана*	*Себя*	*себя*
Творительный	*Хассаном*	*Хассаном*	*Тобой*	*тобой*	*Хассаном*	*Хассаном*	*Собой*	*собой*
Предложный	*Хассане*	*Хассане*	*Тебе*	*тебе*	*Хассане*	*Хассане*	*Себе*	*себе*
Цель увидит								
Именительный	*Ты*	*ты*	*Leo*	*Leo*	*Ты*	*ты*	*Leo*	*Leo*
Родительный	*Тебя*	*тебя*	*Leo*	*Leo*	*Себя*	*себя*	*Leo*	*Leo*
Дательный	*Тебе*	*тебе*	*Leo*	*Leo*	*Себе*	*себе*	*Leo*	*Leo*
Винительный	*Тебя*	*тебя*	*Leo*	*Leo*	*Себя*	*себя*	*Leo*	*Leo*
Творительный	*Тобой*	*тобой*	*Leo*	*Leo*	*Собой*	*собой*	*Leo*	*Leo*
Предложный	*Тебе*	*тебе*	*Leo*	*Leo*	*Себе*	*себе*	*Leo*	*Leo*
Третье лицо увидит								
Именительный	*Хассан*	*Хассан*	*Leo*	*Leo*	*Хассан*	*Хассан*	*Leo*	*Leo*
Родительный	*Хассана*	*Хассана*	*Leo*	*Leo*	*Хассана*	*Хассана*	*Leo*	*Leo*
Дательный	*Хассану*	*Хассану*	*Leo*	*Leo*	*Хассану*	*Хассану*	*Leo*	*Leo*
Винительный	*Хассана*	*Хассана*	*Leo*	*Leo*	*Хассана*	*Хассана*	*Leo*	*Leo*
Творительный	*Хассаном*	*Хассаном*	*Leo*	*Leo*	*Хассаном*	*Хассаном*	*Leo*	*Leo*
Предложный	*Хассане*	*Хассане*	*Leo*	*Leo*	*Хассане*	*Хассане*	*Leo*	*Leo*

help arg

Аргументы большинства команд и заклинаний - вещи и персонажи. Для них есть более или менее универсальный синтаксис.

имя - когда цель - это персонаж или вещь с таким именем. Предметы и живые существа отзываются на все слова своего короткого имени, кроме слов вроде "the" или "of". Короткое имя - это то, которое видно по scan и в бою (для животных) или в equipment/inventory (для предметов). Слова с буквой 'Ё' можно заменять на слова с буквой 'Е': чёрт - черт, тёмный - темный, и т.д.

Также все предметы отзываются на свой тип ('container', 'weapon'), а все живые существа на свою расу ('elf', 'horse'). Эти слова можно сокращать.

Многим не нравится переключать регистр для ввода имени предмета/существа после команды. Для этого почти у всего есть ещё дополнительные имена в виде английских слов или русских в транслите. Первое из этих имён показывается в скобках [], если включены опции config mobengname on и config objengname on.

.имя - точка перед именем означает, что требуется точное совпадение имён.

Например, kill .romi всегда сработает на игрока по имени Romi, но никогда - на моба Romiere's General. Если не указать точку, а рядом будут стоять Romi и Romiere's General, то точное имя будет иметь приоритет: kill romi сработает на игрока, kill romiere - на моба, а kill rom - на того из них, кто окажется в комнате первым. Точка перед именем особенно полезна в ПК и для tell (сам поймёшь, почему).

-имя - чёрточка перед именем означает, что требуется точное совпадение с Истинным Именем Неупокоенного (help undead).

all, все - все предметы, если только команда может действовать с несколькими предметами.

N.имя или N.'имя имя', иногда 'N.имя имя' - N-й по счёту предмет или персонаж с таким именем.

N*имя, N имя, или N*'имя имя' - N штук вещей с таким именем, если только команда может действовать с несколькими предметами. Форма с пробелом после N может применяться только в последнем аргументе команды.

all*имя или all*'имя имя' - все вещи с таким именем, если только команда может действовать с несколькими предметами.

1*gold или 1*золото - 1 золотая монета.

1*coin или 1*silver или 1*серебро - 1 серебряная монета.

Просто gold или silver - это предмет с таким именем.

22*gold - 22 золотых.

all*gold - все золотые монеты.

all*coin или all*silver - все серебряные монеты.

1, 2,... - некоторые команды допускают эту упрощённую форму, имея в виду первого моба в комнате или первый предмет в inventory (например, drop 1).

Ключевые слова:

'o', 'item', 'obj', 'object', 'предмет', 'вещь' - любой предмет.

'm', 'mob', 'npc', 'моб', 'монстр' - любое существо, управляемое сервером.

'p', 'персонаж', 'игрок', 'player' - любой игрок.

'self', 'себя' - ты сам.

'victim', 'жертва' - тот, кого бьёшь.

'carrier', 'car', 'возчик' - тот, на ком едешь.

'pet', 'пет' - приручённое тобой животное.

'tank', 'танк' - тот, кого бьёт тот, кого ты бьёшь.

'boss', 'босс' - хозяин того, кого ты бьёшь.

'i', 'и' - любой предмет в инвентаре (inventory).

'eq', 'эк' - любой предмет экипировки (equipment).

'r', 'к' - любой предмет в комнате.

Ключевые слова сокращать нельзя.

Предметы в equipment отзываются на слово 'eq', в inventory - на 'i', в комнате - на 'r'. Например, "cast bless 1.i".

Некоторые важные замечания:

1) Лишние аргументы часто игнорируются без сообщения об ошибке.

2) Аргументы можно сокращать, используя только начало имени.

3)

Распространённая ошибка:

cast create water - это вовсе не заклинание create water.

MUD понимает это так: первый аргумент 'create'. такого заклинания нет, значит это сокращение, берём первое попавшееся: 'create food'. Этому заклинанию больше аргументов не надо, так что слово 'water' игнорируется. Вот так и получается, что cast create water - это всё равно что cast create или cast 'create food' A cast 'create water' - это правильный вариант.

Время MUD-сервера

Календарь в нашем мире устроен следующим образом: В любом месяце 35 дней, 1-ый день месяца - всегда день луны, 8-й - снова день луны и так далее до 35-го, потом снова день луны нового месяца.

Неделя состоит из 7 дней: Луны, Быка, Лжи, Грома, Свободы, Великих Богов, Солнца

В каждом году 17 месяцев (начиная с нового года):

- [зима] 1. Зимы, 2. Зимнего Волка, 3. Холодного Гиганта
- [весна] 4. Древних Воинств, 5. Великих Битв, 6. Весны, 7.Природы
- [лето] 8. Тщетности, 9. Дракона, 10. Солнца, 11. Жары
- [осень] 12. Битвы, 13. Темноты, 14. Тени, 15. Длинных Теней
- [зима] 16. Абсолютной Темноты, 17. Великого Зла

Времена года даны для северного полушария, в южном полушарии вместо лета - зима, вместо осени - весна, хотя месяцы называются так же.

Годы составляют двенадцатилетние циклы (времена):

1.Бабочки, 2.Мыши, 3.Сурка, 4.Кошки, 5.Змеи, 6.Дракона, 7.Волка, 8.Щуки, 9.Эльфа, 10.Ворона, 11.Льва, 12.Цветка.

Двенадцать времён составляют век:

1. Любви, 2. Возрождения, 3. Жизни, 4. Познания, 5. Молодости, 6. Расцвета, 7. Силы, 8. Зрелости, 9. Старости, 10. Смерти, 11. Печали, 12. Забвения.

Века именуются так: Огня, Воды, Земли, Ветра.

Соотношение течения времени в двух мирах - нашем и "реальном":

Час (тик)	= реальной минуте.	Год	= 10 реальным дням.
День	= 24 реальным минутам.	Время	= 119 реальным дням.
Месяц	= 14 реальным часам.	Век	= 4 реальным годам.

```
function ship_victoria_init(obj)
    -- кэшируем всё что можно в локальные переменные
    local delay, svt, ship_echo, obj_long_descr, obj_to_room = delay, ship_victoria_table,
World.get_room(5400).echo, obj.long_descr, obj.to_room
    local i, room, cmd = 1 -- обьявляем локальные переменные
Mud.log("begin_init")
    while valid(obj) do
Mud.log("init")
        delay(3, obj)
Mud.log("delay3")
        if obj.in_room then -- вдруг какой имм поднял корабль Виктория, тогда in_room == nil!
            cmd = svt[i]
            if cmd == "end" then
Mud.log("end")
                i = 1 -- если цикл дошел до конца, возвращаем индекс на начало
                cmd = "stop"
            end
            if cmd == "stop" then
Mud.log("stop")
                ship_echo("Корабль прибыл в порт ({W"..obj.in_room.area.name.."{x).")
                obj_long_descr = "Красивый {Wбелый корабль{x Виктория стоит у причала."
                   delay(30, obj)
                ship_echo("Корабль отбыл из порта.")
                obj_long_descr = "Красивый {Wбелый корабль{x, скользит по волнам."
            else
Mud.log(cmd)
```

```
                obj_act_to_room(obj, ship_victoria_data_table[cmd]) -- сообщить в зависимости от cmd
куда уплывает корабль (юг/север/запад/восток)
                        room = obj.in_room.exit(cmd)
                        if not room then  -- если вдруг какой-то сбой, то рестартануть корабль с начала
списка
                    i = 0
                                room = 9795
                end
                obj_to_room(room)
                obj_act_to_room(obj, "$p приплывает.")
            end
Mud.log(i)
            i = i + 1
        end
    end
Mud.log("exit?!!")
end
```

нет	число [0-16]	month	Месяц. Месяцы соствляют годовой цикл 1. Зимы, 2. Зимнего Волка, 3. Холодного Гиганта 4. Древних Воинств, 5. Великих Битв, 6. Весны, 7.Природы, 8. Тщетности, 9. Дракона, 10. Солнца, 11. Жары, 12. Битвы, 13. Темноты, 14. Тени, 15. Длинных Теней, 16. Абсолютной Темноты, 17. Великого Зла
нет	число [0-11]	year	Год. Годы составляют двенадцатилетние циклы (времена): 1. Бабочки, 2. Мыши, 3. Сурка, 4. Кошки, 5. Змеи, 6. Дракона, 7. Волка, 8. Щуки, 9. Эльфа, 10. Ворона, 11. Льва, 12. Цветка.
нет	число [0-11].	gener	Время - глобальное значение, общее для всех арий. Двенадцать времен составляют век: Любви, Возрождения, Жизни, Познания, Молодости, Расцвета, Силы, Зрелости, Старости, Смерти, Печали, Забвения.
нет	число [0-3]	cent	Век - глобальное значение, общее для всех арий. Века именуются так: Огня, Воды, Земли, Ветра.

Перечень, описание ошибок компиляции, часто встречающиеся ошибки.

Перечень

<eof> тратата end - если вставить больше концов блоков end, чем начала блоков (function, if, while, for и т.д.)

Ошибки программирования:

зацикливание события SPEECH

```
function handler_event_speech(mob, ch, speech)

    mob.execute(“say кто здесь?”)
end

if mob ~= ch then
    mob.execute(“say кто здесь?”)
end
```