Item Editor Howto

The item editor will allow you to create your magic, items, special items, armors, weapons, and modify the name/stats on the skills. Each one will be covered in a separate section. This guide assumes that you know something about programming. If you know nothing about programming, then this guide will go over you head and leave you dazed and confused. But first, we will cover how the external files are made and stored.

External File makeup:

In this gude, the words “thingies / thingy” will refer to all magic, items, special items, armors, weapons, skills, and spells. All thingies have have 2 arrays. One for the stats of the thingy and one for the name of the thingy. The stat list is an array of 50x30. The name list is 50x15. The rows on each array contain one set of information on any thingy. To keep things simple, the row numbers in the name list and the stat list will be the same for each thingy. For example: lets say that a piece of armor is stored in the stat array at row 4. The name of the piece of armor would also be at row number 4 in the name array.

If you notice, there are 50 rows. This means that you can have up to 50 thingies in each array. There are 30 columns in the stat array. This allows for 30 different stats to exist for any given thingy. However, not all of the columns will be used for every thingy. For example: weapons do damage, so the damage column would not be used for a piece of armor or an item. And vice versa. A weapon or magic would not use the absorb % column that would only be used for armors.

So if all columns are not used for all the different thingies, then why have the wasted space? Simple: the uniformity keeps my code smaller for the main engine and makes things less complicated. And besides, I ran into issues when I tried to cross reference columns with different thingies. So, this fixes the problems.

Each thingy will have its own external files associated with it. The files are as follows: armrX (for armor), weapX (for weapons), itemX (for items), spllX (for spells), skillX (for skills), and spclX (for special items). Each of these will have a name file as well. The name file will have an “n” in front of the name of the file to let you know that it is a name file. For example: narmrX and nweapX are both the name files for the armor list and the weapon list. As mentioned in the game info pdf, you can create up to 1000 different thingies of each thingy. So this raises the question: “how is that possible, if you can only store up to 50 thingies per file?”

This is simple: you create multiple files. The X on the end of the file names represents a number. This can be any number from 1 to 25. So what can you do? You export 50 thingies to a file at a time. For each new file, increase the file number by 1. For example: for the armors: let's say that you have 100 different armors. You export the first file as armr1 and the second file as armr2.

The game engine knows how to handle this and will search for the thingies through all the files until there are no more to search through, or until it finds that individual thingy. The item editor should have coding in it to bypass 50 items per ID number. Meaning that if you type in ID number 2, then you will start with the second set of 50 items. If you start with ID number 4, then it should start around the 150th item.

** Note: the numbers that can be used is 1 through 25. If you go out of that range, then it will not search through that file. **

The array:

The stat array is called item_list. Here is the info on the item_list array:

item_list[i][0]= this is the ID number of the thingy. (the case number should be stored here)
item_list[i][1]= the level of the thingy.
item_list[i][2]= this is for the "type" of thingy, like armor type or weapon type.
item_list[i][3]= the cost of the thingy.
item_list[i][4]= on armors, this will dictate whether you will gain or lose hp.
item_list[i][5]= on armors, this will dictate whether you will gain or lose mana.
item_list[i][6]= on armors, this will dictate whether you will gain or lose str.
item_list[i][7]= on armors, this will dictate whether you will gain or lose dex.
item_list[i][8]= on armors, this will dictate whether you will gain or lose int.
item_list[i][9]= on armors, this will dictate whether you will gain or lose def.
item_list[i][10]= this is how much an armor will absorb.
item_list[i][11]= this is the max str a weapon can handle before it breaks.
item_list[i][12]= this is how much damage a weapon will do.
item_list[i][13]= this is how much fire damage an armor will absorb.
item_list[i][14]= this is how much ice damage an armor will absorb.
item_list[i][15]= this is how much water damage an armor will absorb.
item_list[i][16]= this is how much lightning damage an armor will absorb.
item_list[i][17]=flag 1;
item_list[i][18]=flag 2;
item_list[i][19]=flag 3;
item_list[i][20]=flag 4;
item_list[i][21]=flag 5;
item_list[i][22]= how much a potion will restore.
item_list[i][23]= how long the spell will last on the potion.
item_list[i][24]= how strong the spell is.
item_list[i][25]= how hard it is to learn a skill / spell.
item_list[i][26]= how much mana it costs to cast a spell.
item_list[i][27]= unused. Here for expansion.
item_list[i][28]= unused. Here for expansion.
item_list[i][29]= unused. Here for expansion.

If you want more information, then look below for more information on each thingy type. The array is a 16 bit array, meaning that you cannot surpass the number 32767. Keep this in mind. The cost is a special setup.

Cost:

I enabled the game to have a large cost for thingies exceeding well over 1 million gold. However, the cost of a thingy is limited to a 16 bit number. Therefore, I came up with a solution. The cost of a thingy will be whatever number you decide up to 30,000. After you surpass 30,000, the cost will be dictated by a simple algorithm. It will be (cost-30,000) * 1,500. meaning that a cost of 30,001 will be a total cost of 1,500 NOT 30,001. If you do a cost of 30,834, then the total cost will be 1,250,000.

Flags:

If you notice, there are 5 flags in the array. The flags are there to tell the thingy what kind of effect to have. It will define your magic types, your spells on your items, the added stats on your armors, and what type of attacks your weapons can do. Here is a list on all of your flags:

Armor flags:

perm_haste - this causes your character to always be hasted
perm_sanc - this causes your character to always have sanctuary.
immune_blind - immunity to blind
immune_poison - immunity to poison
immune_silence - immunity to silence
immune_stone - immunity to stone

Weapon flags:

atk_blind - causes the weapon to have a small chance to blind the mob
atk_silence - causes the weapon to have a small chance to silence the mob
atk_stone - causes the weapon to have a small chance to turn the mob to stone
atk_dispel - causes the weapon to have a small chance to dispel the mob
atk_death - causes the weapon to have a small chance to kill the mob instantly.
atk_poison - causes the weapon to have a small chance to poison the mob

Magic and Item flags:

healing – this will restore hp
haste – make the character have an extra hit for a certain number of rounds
Sanc - reduce damage by half for a certain number of rounds
cure_blind - cure blind
cure_poison - cure poison
cure_silence – self explanatory
cure_stone ...
spell_life – brings a character back to life with 1 hp.
full_life - brings a characters back to life with full hp.
mass_heal – heals all characters a certain amount of hp.
cure_mana – only works with items – it restores mana by a given amount.

Dispel - removes maledictions, haste, and sanc if landed.
Blind – this reduces the chances of a hit by about 50% or so.
Poison - each round the player will take damage due to poison.
Silence – the player cannot cast.
Stone - they become turned to stone and you cannot use them until you turn them back to normal.

Spells that hit one creature:

fire – fire damage
ice – self explanatory
water ...
lit - lightening damage

** NOTE: you can use the fire, ice, water, and lit flags to give weapons elemental damage of that type.**

Spells that hit all creatures:

lit_2 – self explanatory
fire_2 ...
ice_2 ...
water_2 ...

Spells that hit randomly:

lit_3 ...
fire_3 ...
ice_3 ...
water_3 ...

** These spells will hit the first target, then will "bounce" off that target and randomly hit another target. This target my be an ally or a foe. This will continue until the spell fizzles out. Technical terms: These are considered "chain" spells. The number of bounces are determined by the level of the player that cast the spell. Basically, there is a loop that will reduce the level by 4 every time it loops. Each loop will cause the spell to "chain," or bounce one time. This will happen until the level is 0 or below. So if a player is level 40, then you will have 10 chains. But each chain is random, so it is a double edged sword. It can kill off your party before it will kill off all of the mobs or vice versa. **

All of these flags are set in defines. Do not change the numbers in the defines, otherwise you could end up with unexpected results. The defines are hard coded in. All these names can be used to set up your thingies. For example you can do the following: "flag1=lit_3;" and this will cause the spell to be of type chain lightening.

Armors:

Armors are probably the most complex piece of equipment (eq) that you can create. There are 4 different types of armor: Helmet, Armor, gloves, and shield. However, each one has the same setup. Armors use 20 different columns of the item_list. ID, level, Hp, Mana, Cost, Str, dex, int, def, type, absorb%, ice absorb, water absorb, lit absorb, fire absorb, and flags 1 to 5.

First is the ID. The ID numbers for armors range from 1000 to 1999. You cannot change the range. It is hard coded in. If you go outside of the range, then it will no longer be an armor. It would be best to use a simple numerical order for your armors. This will keep odd errors from happening.

The Level column will decide what level a character must be before they can use the armor. If you want the armor to be used at any time, then simply make it level 1, as level 0 might cause unexpected errors.

The hp and mana columns will give additional hp or mana to the character wearing the armor. When the armor is removed, so will any extra hp or mana that the armor gives to that character. You could make the number a negative number. If this happens, then the character will lose hp or mana instead of gaining it.

The str, dex, int, and def columns work like the hp or mana. Negative numbers work the same way. ** Warning: do not give high number to the stats. This could greatly unbalance the game. The str works as a multiplier to damage, so a high str will give massive increases in damage dealt. Same thing with int and magic. Not unless the mobs you create are ungodly strong, then there is no need to give excessive amounts of stats to armor. The only exception would be def. Def is helpful in the early game, so giving a couple of pieces of armor with +10 def in the early game could be quite helpful. At low levels, mobs don't do much damage and since each def point reduces the total damage by 1 point, +10 def would reduce total damage by 10 points – always. **

Absorb % works as a percentage damage reduction. An absorb % of 5% would reduce damage by 5%. Keep in mind that all absorption rates for all equipped armors are added together to attain a total absorption rate. So if you have 5% absorption rate on all 4 pieces of armor equipped, then you will have a total of 20% absorption rate. An absorption rate of 100% or more would prevent all damage from being dealt. As a general rule: keep the absorption rate for any piece of armor at 24% or below. This will help to balance the game a bit.

Elemental absorption (ie fire, ice, water, and lit): The elemental absorption runs from column 13 to 16. each column has a different absorption stat for each element. Look at the array above to find out which is which. The elemental absorption process works like the absorb % of damage; however, this is absorption of magic spells. Keep the same general rules of 24% or less for each piece of armor.

Type (column 2): This is for the armor type. This is very important, as it tells where the armor can be equipped. If you fail to input a type into it, then you will not be able to equip the armor. There are 4 different types: helmet, armor, gloves, and shield. There are defines for these, so you can do: “type=helmet;” and it will set the armor type to helmet.

Flags: You can add up to 5 different flags to a piece of armor. Each flag can cause the armor to do something different. The flags you can use are: perm_sanc, perm_haste, immune_blind,

immune_poison, immune_stone, immune_silence. You could add “magic” flags to it, but whether it will have any affect or not is questionable. “Magic” flags are flags that define what type of magic spell a magic spell is. For example, the blind flag might cause the wearer of the armor to be permanently blind. However, I am not 100% sure on this, so if it doesn't work, then it was not implemented to do that. The first 6 flags I mentioned will work, though. Look at the Flags section for more info on the flags.

Weapons:

There are 6 different types of weapons: Dagger, H2H, ninja, exotic, sword, club. The ID numbers for weapons range from 2000 to 2999. You cannot change this – it is hard coded in. The cost column is used for weapons. Read the cost section for more info.

All weapon types correspond to a different skill that a character can learn. Dagger, sword, and club are self explanatory. Ninja is for ninja type weapons like a katana, ninja stars, sais, etc. H2H is for hand to hand combat. You can increase the hand to hand combat damage with special weapons like brass knuckles, claws, etc. Those are the H2H weapons. Exotic weapons are weapons that would not be considered to be a normal weapon, or a weapon type that would not fall under the other weapon types. For example: an 8-ball, a purse, a chair, spray paint, slimy goo, voodoo dolls, flame thrower, wielding torch, etc. Or whatever your imagination can think up. My main idea was to use humorous items as weapons; however, they are a little cheaper than conventional weapons and do more damage.

The Level column will decide what level a character must be before they can use the weapon. If you want the armor to be used at any time, then simply make it level 1, as level 0 might cause unexpected errors.

The weapon damage is used against a multiplier. The multiplier is random and based off of str, too. Basically, do: weapon damage * (str/4) * random (1->17) (based off of characters level) * skill % and you get your damage. Take that into your consideration, or just randomly play around with different weapon damages in order to get a good feel on what your damages should be in order to balance the game. ** Note: the true algorithm is more complex than what was stated above, but that gives a general idea. **

Max str – this tells the weapon at which point it will break. Basically if a character's str is greater than the weapon's max str, then the weapon will break. If you make the max str less than 0 or a negative, then it will become unbreakable. Use -1 as the unbreakable sign, just in case I forgot the logic for lower numbers. Whats good about max str? You can create one use weapons that are very powerful. Like a glass sword. Or a nuke. You know... Something like that.

Weapon flags make the weapon more fun and interesting. There are 6 different attack type flags you can add to the weapon: atk_blind, atk_dispel, atk_death, etc. Look at the Flags section for all of them. You can combine these flags together to make a weapon more dangerous. For example: you can make a weapon called “silent death” and add the flags atk_silence and atk_death to it. That way it will silence the mob and / or kill it. If you wish to make the weapon of an elemental type, then add a magic element type to it, like fire, ice, water, or lit. You can make it of all elemental types if you wish. However, if the mob absorbs lit but is vulnerable to fire, then it will absorb and take damage all at the same time, which might render your weapon ineffective. Keep that in mind.

You can add up to 5 flags to a weapon. Adding any other flags to the weapon, other than the atk flags and the elemental flags may just be a waste of time, because they probably wont do anything.

Items:

These are things that your party can use. You cannot wear them. The ID number range is 3000 to 3999. Items are also level dependent. If you do not understand level dependency, then read the armor section. Read the cost section for how cost works. Items are quite easy to make. You just need the duration or restore amount and the flag(s).

Duration is how long the spell will last in battle (ie how many rounds it will last. A single round is one set of attacks. Each time you get to attack again, the round ends and your duration decreases.) These are good for maledictions, sanc, and haste.

Restore amt is how much the potion will restore to HP or mana. For example, if it has a restore amt of 50, then it will restore only 50. If the restore amt is 1200, then it will only restore 1200. Simple. If the restore amount is greater than your max hp or mana, then it will not surpass your max. This is a basic concept for healing potions that heal one person. For potions that heal all the people, then the restore amount is configured a bit differently.

For mass heal, the restore amount is determined in percentages. Meaning that the mass heal will restore x% of hp and / or mana to each party member. The percentages range from 1% to 100%, where 100% is a full heal and 1% and only 1% of total hp and / or mana.

You can add up to 5 different flags to 1 item. Look at the Flags section for all the different flags you can use. If you decide to use elemental flags on an item, then that spell will be cast upon the user. The spells on items are at spell strength level 25. In reality, you can add 5 lit_3 spells to the item and it would end up like a level 120 lit_3 spell. It would hit that many times. This would probably suicide the party, but eh... who knows.

** NOTE: I know the first item flag will work. As for the others... it is questionable. It does exist inside of a loop, so in theory it should work. If the other 4 flags don't work, then just use flag1.**

Magic:

Magics are very simple to create. The ID number range is 4000 to 4999. They are level dependent. They also have a cost to buy. Magic spells only use flag1. All the other flags are not used. Choose a spell and put it into flag1 and the magic spell will be of that type. Other attributes of spells are Learn, spell str, and mana cost.

Learn (column 25) – determines how hard it will be for a certain spell to increase when practicing it. The lower the number, the easier it is to learn. The higher the number, the more difficult it is to learn. This number will affect how much the percentage rate will increase when you practice the spell in the practice shop. It will also determine how quickly you will learn the spell when using it. You might want to keep the numbers lower than 7. Do NOT use a number lower than 1. This could cause unexpected errors I did not plan for. It could cause an infinite loop, or cause the game to crash. You have been warned.

General rule for the Learn attribute: The more powerful the spell, the higher the learn. The weaker the spell, the lower the number for learn. My most powerful spells I created had a max learn of 5. This made the spells incredibly difficult to learn..

Spell Str acts as a multiplier with int to determine how much damage or healing power the spell has. When it comes down to maledictions, sanc, and haste, the algorithm is quite simple: (spell str*your level)/10= round duration. On sanc, I made a spell str of 3. On my most powerful damage spells, I used a spell str of 125. General rules: on maledictions, sanc, and haste, use low spell strengths, otherwise they will last too long and unbalance the game. On damage spells, use your own discretion, based upon how powerful your mobs are.

Mana cost is special. You want to use a higher mana cost, because as the character's level goes past the spell's level, the mana cost for that spell will drop. Eventually, when you attain enough level difference between the spell's level and the character's current level, the mana cost will be 5 mana. This is for any spell you create. So if the spell is really powerful, make it a high level and make sure the mana cost is outrageous. Look at the mana costs for the spells I created. This might give you an idea on what you could do.

Skills:

All you can really do to skills is rename them, reprice them, and adjust their learn. If you change their ID's you will screw things up. General rule: leave the skills alone and don't mess with them.

Special items:

These are items that do absolutely nothing. They have no abilities whatsoever. They have a cost, a level, and an ID. The ID ranges from 9000 to 9999. The purpose of special items is to help aid the storyline. For example: you may need a key to unlock a door, so you will create a special item (called “key”) and you will do a check, using AFWSL, to see if it exists in the inventory. If it does, then you will progress to wherever. That is the point of special items. They could have more uses in the future. Late in the game making process, I made things a bit more scalable, just in case I wanted to do something else, like customize weapons/armor in game.

Item Creation:

Now that you have a general idea on what you can make, we are going to go in and explain how to use the item creator to make items. Now let's have some fun coding. But first... we have to learn how it is set up.

Basics:

You have a huge list of defines to make things easier. Here is a list of them. They can also be found in items.h:

```
// spells
 #define healing 1
 #define haste 2
 #define sanc 3
 #define cure_blind 4
 #define cure_poison 5
 #define cure_silence 6
 #define cure_stone 7
 #define spell_life 8
 #define full_life 9
 #define mass_heal 10
 #define cure_mana 11

 #define dispel 30
 #define blind 34
 #define poison 35
 #define silence 36
 #define stone 37

// elements
#define fire 8001
#define ice 8000
#define water 8003
#define lit 8002

#define lit_2 8004 // this is for hit all.
#define fire_2 8005// the <ele>_2 is for the hit all
#define ice_2 8006
#define water_2 8007

#define lit_3 8010// this is the chain lit affect. - for the other eles, use the 801x #s
#define fire_3 8011
#define ice_3 8012
#define water_3 8013

// armor types
#define helmet 1
#define armor 2
#define gloves 3
#define shield 4
#define weapon 5

// weapon types
#define dagger 11
#define sword 12
#define club 13
#define h2h 14
#define ninja 15
#define exotic 16
```

```
// skill types
#define h2hc  5000 // h2h combat
#define exoticw 5001
#define daggers 5002
#define clubs 5003
#define swords 5004
#define atk2 5005 // 2nd attack
#define ninjitsu 5006
#define atk3 5007 // 3rd attack
#define enh_w_dam 5008 // enhanced weapon damage
#define enh_m_dam 5009// enhanced magic damage
#define shield_block 5010
#define dodge 5011

#define perm_haste 10000
#define perm_sanc 10001
#define immune_blind 10002
#define immune_poison 10003
#define immune_silence 10004
#define immune_stone 10005
#define atk_blind 10006
#define atk_silence 10007
#define atk_stone 10008
#define atk_dispel 10009
#define atk_death 10010
#define atk_poison 10011
```

All of these defines are very useful when it comes down to item creation. Instead of memorizing the stupid numbers for each flag and item types, you can simply use the word instead. It made it easier on my life. DO NOT modify the numbers of the defines.

All of the items are defined in one huge case statement in the item_list.c file. Each case represents the ID number of the item. Within that case, the item is defined and created. For example, let's look at some code:

Armor creation:

```
case 1112:  - this is the id number
// fire shield
        absorb=18;  - this is the absorb% - try not to go over 24%
        type=shield; - this initializes the armor type
        hp=100;  - this gives a hp bonus
        mana=100;  - this gives a mana bonus
        fire_res=25;    - this absorbs 25% of fire attacks.
        Level=70;  - the level of the shield
        cost=30034;  - the cost is 34*1500 – see cost section for more details.
        strcpy(iname,"Fire Shield");  - this is how you create your name for the item
break;
```

When you look at it, you only need to define those things, which you want to have on the armor. By default, when the program is cycling through the loop, it resets all values back to 0. If the value is 0, then it does not have that stat. This made it easier to make items, by defining only what you needed.

** NOTE: the name of any thingy is limited to 12 characters, including spaces. Go over this limit and you will cause problems for the game.

Weapon creation:

```
case 2108:  - id number
//fire Sw
        dmg=250;  - the damage multiplier
        mstr=500;  - max str of 500, which will probably never be reached, so it may never break.
        type=sword;  - weapon type is sword
        level=70;  - level is 70
        cost=30133;  - cost of the item is 200000;
        flag1=fire;  - give it fire damage
        strcpy(iname,"Fire Sword");   - the name of the weapon

break;
```

Item creation:

```
case 3000:  - id number
        // haste  - a comment that lets me know what the item is – good practice to do this
        cost=100;  - cost
        level=1; - level
        flag1=haste;  - type of spells it has (haste)
        duration=3;  - how many battle rounds it will last
        strcpy(iname,"Energy Pill"); - name of item
break;
```

Magic Creation:

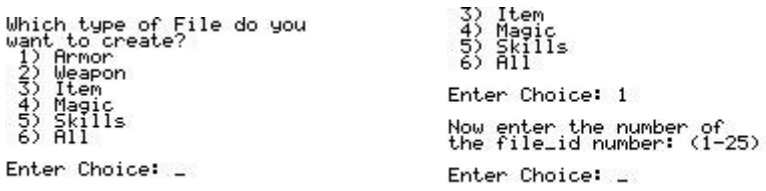
```
 case 4011:  - id number
// arch light - hits all
        learn=1;  - learning difficulty (very easy)
        spell_str=7;  - spell str – not very strong
        mana_cost=250;  - originally costs 250, but will go down as you level
        level=15;  - must be level 15 before you can learn it.
        cost=1500;  - cost 1500 to learn it.
        flag1=lit_2;  - type of magic spell (hit everything lit spell)
        strcpy(iname,"Arc Lghtning"); -name of spell

break;
```

Skill Creation:

```
case 5001:  - id
//exotic
        learn=2; - learn rate
        level=1; - level you must be to learn it.
        Cost=3500;  - cost to learn it
        strcpy(iname,"Exotc Weapns"); - name of skill
break;
```

** Remember: you cannot create new skills. You can only modify them. General rule again: don't mess with the skills. Leave them alone. **

Ok now we seen a tiny bit of the case statement. The first thing in weapon creation is figure out what kind of thingy you want to make and prepare the stats for it. Then add it to the case statement in the correct area, by using the general rules set in the previous sections. After that, compile it. Then send it to your calculator and run the editor. When you run it, it will look like the following:



You will have a menu of choices of which thingy to export. After you enter your choice, you will have to choose which file id number to give it. It will be best to start with the number 1. If you surpass 50 of any thingy, then increase the file ID by 1 and continue on this way. That way you can avoid problems.

That is it. You are now done and have created at least one set thingies. You do have the ability to export all thingies at the same time, if you wish. All thingies should be stored under the items folder. If you do not store the files under the items folder, then the game engine will not find them. Keep in mind that you will need to archive everything.

All folders will be created if they do not exist if you run this program. The folders will also be created in the event you run the afw engine and you do not have them.

File list:

itemedit.c – main routine.
map_saver.c – saves the files
item_list.c - contains the main case statement for all of the items.
items.c - routine that fills the arrays with info.
mapping_sys.h – access to all the important routines.
items.h – contains the defines, variables, and arrays – don't edit this.

** TIP: when creating your thingies: put them in the order that they will show up in your different shops, because the shops use a range of ID numbers for what will be displayed in the shops. **