Game Server HOWTO

Anders Jensen−Urstad

<anders@unix.se>

$Id: Game−Server−HOWTO.sgml,v 1.5 2003/04/08 20:49:11 andersju Exp $

This document explains how to install, configure and maintain servers for various popular multiplayer games.

Table of Contents

Table of Contents

1. Introduction

This document describes how to install, configure and maintain dedicated servers for various popular multiplayer games, such as the Quake series, Half−Life/Counter−Strike and other first−person shooters. Linux makes a great operating system for game servers because of its stability and speed (not to mention it's Free).

1.1. Copyright and License

1.2. History

I wrote this document because I couldn't find any decent documents covering more than one game. Wading through piles of readmes, guides, howtos and webpages can be quite tedious.

Revision History

Revision 1.0	2002−12−16	Revised by: aju
Added KTeams Pro to QW. Updated most version numbers and links. Removed the actual GFDL from the document. Many minor changes.		
Revision 0.99	2001−07−08	Revised by: aju
Changed license, minor fixes.		
Revision 0.23	2001−03−03	Revised by: aju
Revised and changed Half−Life/Counter−Strike.		
Revision 0.22	2001−01−28	Revised by: aju
Added Rocket Arena 3 and Alliance to Quake3.		
Revision 0.22	2001−01−26	Revised by: aju
Added Security and permissions, revised QuakeWorld; added QuakeForge. Revised LMCTF. Added Lithium II, L−Fire DM and L−Fire CTF. Revised Q2Admin.		
Revision 0.21	2001−01−15	Revised by: aju
Updated Half−Life/Counter−Strike. Added Rocket Arena to QuakeWorld.		
Revision 0.20	2001−01−14	Revised by: aju
First draft.		

1.3. New versions

The newest version of this document can be found at its homepage http://x.unix.se/howto/ (both HTML and its SGML source). Other versions may be found in different formats at the Linux Documentation Project.

1.4. Credits

I've gathered information from lots and lots of different READMEs, HOWTOs, web pages, and of course personal experience. The Unreal Tournament section was written by Knight Walker <kwalker@aros.net>.

- Christer <vendor@nfn.net> helped with some parts about QuakeWorld.
- The Quake−HOWTO by Bob Zimbinski <bobz@mr.net> gave info about various things.
- The Q2 Server FAQ.
- http://www.atomicage.com:80/quake/server/server_cfg/.
- http://www.planethalflife.com/server/.
- Lots of Counter−Strike info from http://server.counter−strike.net.
- #nucdawn@EFnet, unix.se, #unix.se@freenode
- id Software for making the best multiplayer games.
- Dave "Zoid" Kirsch for making Threewave CTF for all the Quake games and for the Linux ports!
- Loki Entertainment Software for their work on Quake3 and for everything they contributed to the Linux and open−source community.

1.5. Feedback

If you have any questions, comments, suggestions, corrections or whatever, please mail them to <anders@unix.se>.

2. Basics

Game servers can consume a lot of CPU and bandwidth (depending on the game and the number of connected players). If you don't own the machine and want to run a server from your account, ask the system administrator first.

2.1. Security and permissions

All dedicated servers are strongly recommended to be run from another user than root. I recommend that you create a new user that handles all the game servers. You may not have permission to create certain directories mentioned in this document as a normal user, for example `/usr/local/games/quake3`. If so, create it as root and then `chown user:group /usr/local/games/quake`, where user is your username and group your group, or simply create it in your home directory.

2.2. Keeping the server running

If your game server crashes, a shell script like the one below might come in handy so you won't have to restart it manually. It can easily be modified for whatever server(s) you're running.

```
#!/bin/sh

quake3dir="/usr/local/games/quake3"
process=`ps auxw | grep linuxq3ded | grep −v grep | awk '{print $11}'`

if [ −z "$process" ]; then

  echo "Couldn't find Quake3 running, restarting it."
  cd "$quake3dir"
  nohup ./linuxq3ded +exec ffa.cfg &
  echo ""

fi
```

Put the script somewhere, name it sv_up or whatever you like, and make cron run it every 5−10 minutes:

```
*/10 * * * * /usr/local/games/quake3/sv_up.sh >/dev/null 2>&1
```

Put this in crontab (crontab −e). It will execute sv_up.sh (the shell script above) every 10 minutes and its output is sent to /dev/null (in other words, it disappears instead of being mailed to you).

3. QuakeWorld

QuakeWorld is a multiplayer−only version of Quake with optimized network code. id Software released the source code for Quake/QuakeWorld in 1999, spawning a number of projects set out to improve, optimize and add more features to the code and make it more secure. QuakeForge is quite well−developed on the client side, but mvdsv (part of the qwextended project) seems to be the most popular server in the QW community. It's famous for its ability to record demos from each player's POV (Point of View) and store them for retrieval by the players.

3.1. System requirements

Here are the minimum system requirements for the QW (QuakeWorld) server. Remember that the requirements vary depending on the number of clients.

- Pentium 90 or better
- 16 MB RAM
- The *.pak files from the Quake CD−ROM
- Kernel 2.0.24 or later
- Enough free space on your HDD (depends on what you want to install; at least 40−70 MB)

3.2. Installing

id Software's old qwsv is quite dated and should not be used (nor *can* it be used in most cases, since it's built for the ancient libc5). So, use either mvdsv or QuakeForge.

Note: Both mvdsv QuakeForge are actively being developed (more or less), and it's highly possible that newer versions than the ones linked below have been released, so check http://qwex.does.it/ or http://quakeforge.net/files.php before you proceed further.

- mvdsv 0.165b source
- mvdsv 0.165b binary (glibc 2.1.3)
- QuakeForge 0.5.2 source

Create a directory for the QW server in your home directory:

```
$ mkdir quake
```

If you downloaded the mvdsv source:

```
$ tar zxvf qwex_src−0.165b.tar.gz
$ cd mvdsrc
$ make build_sv
$ cp releasei386−glibc/mvdsv ../quake
```

If you downloaded the mvdsv binary:

```
$ gunzip mvdsv.0.165b.linux.glibc2.1.3.gz
$ mv mvdsv.0.165b.linux.glibc2.1.3 mvdsv
$ chmod +x mvdsv
$ mv mvdsv quake/
```

If you downloaded the QuakeForge source:

```
$ tar zxvf quakeforge-0.2.99beta6.tar.gz
$ cd quakeforge-0.2.99beta6
$ ./configure --prefix=$HOME/quake --bindir=$HOME/quake; make; make install
```

This will *at least* build qf−server, nq−server (RealQuake) and some client binaries. We don't need the latter; you can choose not to compile them, or simply remove them later. Anyway, if the compilation went fine you should now have a binary called qf−server in `$HOME/quake`, where $HOME is your home directory (usually /home/user).

You might want to rename mdsv or qw−server to qwsv, since that's what I'll call it in the rest of this document. Now create the directory `id1` in the quake directory and copy the pak0.pak and pak1.pak files from your Quake CD or wherever you have them into that directory:

```
$ cd ~/quake
$ mkdir id1
$ cp /some/where/id1/pak*.pak id1
```

You also need the qw files:

- qwsv230.zip

Unzip it in your `quake` directory:

```
$ cd ~/quake
$ unzip qwsv230.zip
```

Now you're ready to run qwsv. Start it by running `./qwsv` in the quake directory. It should work fine (if not, make sure all the filenames are in lower−case). Try connecting to your server with a client.

3.3. Configuring

Now it's time to configure your QW server. Make a server.cfg file in the id1 directory containing the following:

```
sv_gamedir qw
deathmatch 1
hostname                "QW testserver"
serverinfo admin        "webmaster@xyz.com"
serverinfo url          "http://url.net"
rcon_password xxxx
timelimit 35
fraglimit 150
noexit 1
pausable 0
samelevel 2
maxclients 16
map dm3
floodprot 4 8 30
floodprotmsg "You have activated the flood protection and will be silenced for 30 seconds"
maxspectators 2
allow_dowload 1
allow_download_skins 1
```

```
allow_download_models 1
allow_download_sounds 1
allow_download_maps 1
```

As you can see the server.cfg file contains all kinds of variables the server uses.

- `-master IP-address` − Command line parameter. Connects to the specified master server. QWSV starts in masterless mode by default.
- `-port` − Specifies which port the server will listen to, default is 27500.
- `gamedir` − Which game directory you want to use. Change it if you want to run a mod. The QW directory is used by default.
- `deathmatch` − Can be set to 1, 2 or 3. 1 = Normal deathmatch; Weapons/items can be picked up and respawned (30 sec respawn time). 2 = Weapon stay. You can only pick up a weapon once, ammo & armor doesn't respawn. 3 = Combination between 1 and 2. You can only pick up a weapon once, ammo respawns after 15 seconds, everything else respawns normally.
- `hostname` − Servername.
- `serverinfo admin` − The admin's e−mail address.
- `serverinfo url` − Server URL.
- `timelimit` − Match ends when timelimit (specified in minutes) is reached.
- `fraglimit` − Match ends when fraglimit is reached.
- `maxclients` − Max number of players.
- `map mapname` − Map that will be played.
- `maxspectators` − Max number of spectators.
- `password` − Password protect the server. Clients must set a matching password to be able to connect.
- `rcon_password` − Password used for remote administration.
- `allow_download` − Set this to 1 to allow clients to download files they don't have from the server.

```
$ ./qwsv > /dev/null &
```

The above command runs the qwsv server in the background and sends the output to /dev/null (if you want to log the output, just replace /dev/null with /blah/qw.log or whatever). −port specifies the port the server will use; QW's default is 27500. For a complete list of commands and the official QW manual, see QuakeWorld.net. For a list of special commands that can be used with mvdsv, see the qwex readme.

3.4. Threewave CTF (Capture The Flag)

Capture The Flag, or CTF for short, is − or was, at least − the most popular Quake modification. There are many different CTF variants. Threewave CTF is the original and most popular CTF modification for QW. You need the following files:

- ftp://ftp.sunet.se/planetquake/threewave/ctf/server/3wave42.zip − All the server files.
- ftp://ftp.sunet.se/planetquake/threewave/ctf/server/3wave421.zip − Some bugs fixed (4.2 is required).
- ftp://ftp.sunet.se/planetquake/threewave/ctf/client/3wctfc.zip − All the CTF maps.

Create a ctf directory and extract all the files you downloaded to it:

```
$ cd ~/quake
$ mkdir ctf
$ unzip 3wave42.zip -d ctf
$ unzip 3wave421.zip -d ctf
$ unzip 3wctfc.zip -d ctf
```

Now try to start the server (if you don't specify a map it'll default to an interesting modified version of Quake's start level):

```
$ ./qwsv +gamedir ctf +map ctf1
```

For more information, read `ctf/server.txt`.

3.5. Kombat Teams Pro

Kombat Teams is a very popular mod that's been around for a long time. Most QW servers run it. KTeams greatly simplifies all forms of DM games by supplying commands to set teams, timer, readiness and much more.

Download the following:

- http://www.wsb.poznan.pl/~pawel/q/q/ktpro/ktpro.1.57.tar.gz

Extract it in your quake directory:

```
$ tar zxvf ktpro.1.57.tar.gz
```

Start the server:

```
$ ./qwsv +gamedir ktprosrv
```

Default is a 1on1 server. To start a 2on2 server you'd add +exec 2on2.cfg, or +exec 4on4.cfg for a 4on4 server, or +exec free.cfg for a free for all server. KTPro comes with a bunch of scripts which do this; ktpro1on1, ktpron2on2, etc. in the quake directory. Check out all the configuration examples in `ktprosrv/` and the documentation in `ktpro.doc/`.

3.6. Rocket Arena

Rocket Arena is a very exciting modification. It's one−on−one games with the simple rule "winner stays, loser goes". Each player waits for his/her turn to fight in the arena. Every player gets full armor (200), 100 health and all weapons when they enter the arena. The winner stays to fight again, the loser goes back to the line. Simple? Yes. Boring? Not at all!

Now, on to installing this modification. Get the following files:

- ftp://ftp.sunet.se/pub/games/PC/idgames2/planetquake/servers/arena/fasrv12.zip − All the server files.
- ftp://ftp.sunet.se/pub/games/PC/idgames2/planetquake/servers/arena/farena12.zip − The required client files (maps and sounds).

Create a directory called `arena` in your quake directory and unzip the above files to it:

```
$ cd quake
$ mkdir arena
$ unzip ~/fasrv12.zip −d arena
$ unzip ~/farena12.zip −d arena
```

Start the server:

```
$ ./qwsv +gamedir arena +setmaster 204.182.161.2 +exec rotate.cfg +maxclients 6 +timelimit 20 +fr
```

For the map rotation you can choose one of the following (of course you can edit these or make your own, remember that the last map must loop to the first):

- `rotate.cfg` − All the Final Arena maps.
- `newmaps.cfg` − The new Final Arena and the TF arena maps.
- `classic.cfg` − The most popular older Arena maps.

4. Quake II

Quake II is the sequel to Quake, featuring great multiplayer capabilities just like its prequel.

4.1. System requirements

As always, the minimum system requirements vary depending on the number of clients.

- A Pentium−class processor or better is recommended. More players = more CPU.
- At least 16 MB RAM. 1 MB per player is recommended.
- Some maps require more CPU/RAM than others.
- The *.pak files from the Quake II CD−ROM.
- Enough free space on your HDD, at least ~500 MB.
- If you're going to run a Q2 server on the Internet, make sure you have enough bandwidth. The more players the more bandwidth you need.

4.2. Installing

To start a Q2 server you need the Q2 client package and the pak files from the Q2 CD. Download the one that suits your system (glibc, unless your distribution is really ancient):

- ftp://ftp.sunet.se/pub/pc/games/idgames2/idstuff/quake2/unix/quake2−3.20−glibc−i386−unknown−linux2.0.tar
- ftp://ftp.sunet.se/pub/pc/games/idgames2/idstuff/quake2/unix/quake2−3.20−i386−unknown−linux2.0.tar.gz

Create a directory for Q2 and extract the Q2 client package to it:

```
$ mkdir quake2
$ tar zxvf quake2-3.20-glibc-i386-unknown-linux2.0.tar.gz -C quake2
```

Now create a baseq2 directory inside the quake2 directory and copy the pak files to it from the Q2 CD−ROM:

```
$ mkdir quake2/baseq2
$ cp /mnt/cdrom/baseq2/*.pak quake2/baseq2
```

Go to the Quake2 directory and start the server:

```
$ cd quake2
$ ./quake2 +set dedicated 1 +set deathmatch 1
```

It should work fine and people should be able to connect to it.

4.3. Configuring

You'll want to create a server config (cfg) file. Put the following in a file called server.cfg in the baseq2 directory and modify it as you like:

```
set hostname "Q2 testserver"
set deathmatch "1"
set rcon_password "xxxx"
set timelimit "30"
set fraglimit "100"
set maxclients "16"
set allow_download "1"
set allow_download_players "0"
set allow_download_models "1"
set allow_download_sounds "1"
set allow_download_maps "1"
```

Settings:

- `dmflags` − A bitflag, controls gameplay options.
- `fraglimit` − Next map is loaded when fraglimit is reached.
- `timelimit` − Next map is loaded when timelimit is reached.
- `map` − Which map to start.
- `maxclients` − Max number of players.
- `hostname` − Name of the server.
- `deathmatch` − If you want to play DM (deathmatch), set this to 1.
- `game` − Which mod directory to use, if you want to play a mod.
- `port` − Which port you want the server to listen to.

Now start the server with `./quake2 +set dedicated 1 +exec server.cfg`. Your config file will be read and the variables in it will be used. You can have different config files for different game types, for example ffa.cfg, ctf.cfg, etc. with customized settings. They should all be in the baseq2 directory.

To have your server listed on the master servers (so people using GameSpy, XQF and similar programs can see your server), type `set public 1` in the server console or in the config file. With the command `set setmaster 'master server'` you can change master server (default is id Software's master server).

4.4. Q2CTF (Capture The Flag)

Download one of the following (you'll most likely want the glibc one, unless your distribution is ancient).

- ftp://ftp.sunet.se/pub/pc/games/idgames2/idstuff/quake2/ctf/unix/q2ctf150−glibc−i386−unknown−linux.tar.gz
- ftp://ftp.sunet.se/pub/pc/games/idgames2/idstuff/quake2/ctf/unix/q2ctf150−i386−unknown−linux.tar.gz

It only contains the file `gamei386.so`. Create a ctf directory in your Q2 directory and extract the q2ctf archive:

```
$ mkdir quake2/ctf
$ tar zxvf q2ctf150-glibc-i386-unknown-linux.tar.gz -C quake2/ctf
```

You'll also want the Q2CTF package with the CTF maps, ftp://ftp.sunet.se/pub/pc/games/idgames2/idstuff/quake2/ctf/q2ctf150.zip (9.1 MB). Unzip it to the ctf directory. To start a CTF server, run `./quake2 +set dedicated 1 +set game ctf +exec server.cfg`, where server.cfg is a file you should create in the ctf directory, containing this:

```
deathmatch 3
maxclients 12
rcon_password password
fraglimit 0
```

```
timelimit 30
set hostname "Q2CTF Testserver"
set admin "admin@xyz.com"
map q2ctf1
```

4.5. LMCTF (Loki's Minions CTF)

LMCTF, Loki's Minions CTF, is a popular CTF modification for Q2. You need all the following files:

- ftp://ftp.sunet.se/planetquake/lmctf/lm520−linux_bin.zip
- ftp://ftp.sunet.se/pub/games/PC/idgames2/planetquake/lmctf/lmctf456.zip
- ftp://ftp.sunet.se/pub/games/PC/idgames2/planetquake/lmctf/lmctf50.zip

Create a `lmctf` directory in your Q2 directory and unzip all the files to it. Go to the Q2 root directory and start with the following command:

```
$ ./quake2 +set dedicated 1 +set deathmatch 1 +set game lmctf +exec server.cfg
```

There are some files in the lmctf directory that you'll probably want to edit: `server.cfg` is the main cfg file containing many standard server options. `maplist.txt` contains the list of maps the server cycles through. To alter the map list, create a file in your root Q2 directory called maplist.txt containing the names of the maps and add +map mapname to the command line when you start the server, where `mapname` is the name of the first map in the maplist.txt you just created. The file `motd.txt` contains the "message of the day" clients will see when joining the server.

For more information on LMCTF, see its homepage.

4.6. Rocket Arena 2

To run a Rocket Arena 2 server you need the following files:

- ftp://ftp.sunet.se/pubi/pc/games/idgames2/planetquake/servers/arena/ra2250sv.zip
- ftp://ftp.sunet.se/pubi/pc/games/idgames2/planetquake/servers/arena/ra2250cl.exe − required client files (maps and such).

Create a directory called `arena` in your quake2 directory and extract the above files:

```
$ cd quake2
$ mkdir arena
$ unzip ra2250sv.zip −d arena
$ unzip ra2250cl.exe −d arena
```

Edit `server.cfg` to your liking. Another file that you may want to edit is `arena.cfg`, which is used to customize settings on a per arena basis as well as map rotation information. Start the server:

```
$ ./quake2 +set dedicated 1 +set game arena +exec server.cfg
```

4.7. Lithium II

Lithium II is a very popular and configurable server−side deathmatch mod, adding things like the grappling hook and letting you configure pretty much everything; see its readme.txt file for more information. Download the following file:

- ftp://ftp.sunet.se/pub/games/PC/idgames2/planetquake/lithium/lithium2_1.24−i386−unknown−linux2.0.tar.gz

Extract the file in your quake2 directory. All the files will be placed in a new directory called `lithium`:

```
$ cd quake2
$ tar zxvf lithium2_1.24−i386−unknown−linux2.0.tar.gz
```

Lithium II comes with four different config files, copy one of them to `server.cfg` and modify it to your liking:

- `lithium.cfg` − default Lithium II server.
- `stock.cfg` − stock Quake2 server.
- `lithctf.cfg` − default Lithium II CTF server.
- `stockctf.cfg` − stock CTF server.

Start a deathmatch server with the following command:

```
$ ./quake2 +set dedicated 1 +set game lithium +exec server.cfg
```

To start a CTF server, you must have Q2CTF installed. Extract the lithium archive to your ctf directory and start with the following command:

```
$ ./quake2 +set dedicated 1 +set game ctf +set ctf 1 +exec server.cfg
```

4.8. L−Fire DM

L−Fire DM is a server−side mod that adds many features to Q2, such as organized match support, anti−spam, highscore lists for each level, profanity filtering, etc. What you need:

- ftp://ftp.sunet.se/pub/games/PC/idgames2/planetquake/lfire/LFireDM_v1_11_Linux.tar.gz
- ftp://ftp.sunet.se/planetquake/lfire/LFireDM_v1_11_Config.tar.gz

You can run L−Fire DM from either the `baseq2` directory (in that case it'll show up as a standard DM mod in GameSpy and similar tools, or you can give it its own directory (`lfiredm` or whatever you prefer). Extract the archive (actually, it only contains two files, gamei386.so and readme.txt), or in other words place gamei386.so in your directory of choice.

```
$ cd quake2
$ mkdir lfiredm
$ tar zxvf LFireDM_v1_11_Linux.tar.gz −C lfiredm
```

`LFireDM_v1_11_Config.tar.gz` contains many config files which you may want to edit. Extract it to the directory where you put gamei386.so:

```
$ tar zxvf ~/LFireDM_v1_11_Config.tar.gz −C lfiredm
```

Start the server:

```
$ ./quake2 +set dedicated 1 +game lfiredm
```

4.9. L−Fire CTF

L−Fire CTF is like L−Fire DM a server−side mod that adds many features to Q2CTF like match support, anti−spam, highscore lists for each level, rocket arena/sudden death overtime, etc. What you need:

- ftp://ftp.sunet.se/pub/games/PC/idgames2/planetquake/lfire/LFireCTF_v1_20_Linux.tar.gz
- ftp://ftp.sunet.se/pub/games/PC/idgames2/planetquake/lfire/LFireCTF_v1_20_Config.tar.gz

Extract the archive to your Q2CTF directory:

```
$ cd quake2
$ tar zxvf LFireCTF_v1_20_Linux.tar.gz −C ctf
```

`LFireCTF_v1_20_Config.tar.gz` contains a bunch of configuration files which you may want to edit. Extract it to the Q2CTF directory:

```
$ tar zxvf LFireCTF_v1_20_Config.tar.gz −C ctf
```

Start the server just as you normally would:

```
$ ./quake2 +set dedicated 1 +game ctf
```

4.10. Q2Admin

Q2Admin is a very good transparent proxy modificatiom that adds many admin functions, and the most important, ZBot/Ratbot (among other things) detection to preven cheating. Q2Admin works with all Q2 mods transparently by filtering communication between the server and the mod it's running on top of. I recommend that you install Q2Admin if you're going to run a public server. Download this file:

- ftp://ftp.sunet.se/planetquake/q2admin/q2admin18src.tar.gz

Now extract the archive and compile it:

```
$ cd quake2
$ mkdir q2admin
$ tar zxvf q2admin18src.tar.gz −C q2admin
$ cd q2admin; make
```

Now you should have a file called `q2admin.so` in the q2admin directory. For each mod you want to protect with Q2Admin, copy the `install` and `q2admin.so` files to each mod directory and optionally the *.txt files if you want to customize the Q2Admin settings for different mods. Run install, `./install` (chmod +x install if it's not executable) and start the server as usual. The install script moves gamei386.so to gamei386.real.so and q2admin.so to gamei386.so. Run install again to move the files back to their original names. There's a huge number of commands; See `readme.txt` included in the q2admin package for everything you need to know.

5. Quake III Arena

Quake III is the latest game in the Quake series, designed as a multiplayer deathmatch game.

5.1. System requirements

As always, the system requirements vary depending on the number of players on your server.

- Pentium II 266MHz. More CPU = more players.
- At least 64 MB RAM.
- Kernel 2.2.9 or higher, glibc 2.1.
- The *.pk3 files from the Quake III Arena CD−ROM.
- At least ~500 MB free space on your HDD.
- Enough bandwidth if you're going to run an Internet server. More players = more bandwidth.

5.2. Installing

First download the latest Q3A Linux point release. As of this version of the HOWTO it's v1.32b, but that may have changed when you read this.

- ftp://ftp.sunet.se/pub/pc/games/idgames2/idstuff/quake3/linux/linuxq3apoint−1.32b.x86.run

Run the installer:

```
$ sh linuxq3apoint-1.32b.x86.run
```

The default installation directory is `/usr/local/games/quake3`. Copy the *.pk3 files from the Q3A CD−ROM to the baseq3 directory.

```
$ cp /mnt/cdrom/Quake3/baseq3/*.pk3 /usr/local/games/quake3/baseq3
```

Go to the Q3 directory and test the dedicated server, `./q3ded`.

5.3. Configuring

Now it's time to write some config files. They contain all the variables the server will use (maps to be played, gametype, etc..). All cfg's must be in the baseq3 directory. The file q3config.cfg is *always* executed. I recommend that you have different cfg's for different gametypes, for example ctf.cfg, ffa.cfg, and so on. You can use q3config.cfg for general settings, and then another cfg on top of it. Run `./linuxq3ded +exec configfile.cfg` to start the dedicated server and execute the specified config file. This is what my FFA (Free For All) config file looks like:

```
set sv_hostname "Foofighters FFA DM"
set sv_maxclients 10
set g_motd "To be or not to be..."
set g_forcerespawn 15
set rconpassword "password"
set g_gametype 1
set fraglimit 50
set timelimit 20

//Here's the map-cycle. When fraglimit or timelimit is reached, the map is automatically changed.
//Otherwise it would just play the same map again.
set m1 "map q3dm1; set nextmap vstr m2"
set m2 "map q3dm2; set nextmap vstr m3"
set m3 "map q3dm3; set nextmap vstr m4"
set m4 "map q3tourney1; set nextmap vstr m5"
set m5 "map q3dm4; set nextmap vstr m6"
set m6 "map q3dm5; set nextmap vstr m7"
set m7 "map q3dm6; set nextmap vstr m8"
set m8 "map q3tourney2; set nextmap vstr m9"
set m9 "map q3dm7; set nextmap vstr m10"
set m10 "map q3dm8; set nextmap vstr m11"
set m11 "map q3dm9; set nextmap vstr m12"
set m12 "map q3tourney3; set nextmap vstr m13"
set m13 "map q3dm10; set nextmap vstr m14"
set m14 "map q3dm11; set nextmap vstr m15"
set m15 "map q3dm12; set nextmap vstr m16"
set m16 "map q3tourney4; set nextmap vstr m17"
set m17 "map q3dm13; set nextmap vstr m18"
set m18 "map q3dm14; set nextmap vstr m19"
set m19 "map q3dm15; set nextmap vstr m20"
set m20 "map q3tourney5; set nextmap vstr m21"
set m21 "map q3dm16; set nextmap vstr m22"
set m22 "map q3dm17; set nextmap vstr m23"
set m23 "map q3dm18; set nextmap vstr m24"
set m24 "map q3dm19; set nextmap vstr m25"
set m25 "map q3tourney6; set nextmap vstr m1"
```

- `sv_maxclients` − Max number of players.
- `g_motd` − The message people will see on the bottom of the screen when they connect.
- `g_forcerespawn` − Number of seconds until a client is automatically respawned, if the client doesn't do it by itself. Set it to 0 to disable force respawn.
- `g_gametype` − Gametype. 1 = DM, 2 = Tourney (1on1), 3 = Team DM, 4 = CTF.

Start the server with `./q3ded +exec configfile.cfg`. You can execute cfg's directly from the server console with the command `exec configfile.cfg`. If you want to run the server in the background, immune to hangups, start using `nohup ./q3ded +exec ffa.cfg &`.

5.4. Q3CTF (Capture The Flag)

CTF is built into Q3A (Q3CTF). Four CTF maps are included with Q3A, but you will want to download Dave 'Zoid' Kirsch's (the author of CTF for Q1/Q2/Q3) excellent Q3WCTF maps − ftp://ftp.sunet.se/pub/pc/games/idgames2/planetquake/mappacks/q3wctf.zip (7.8 MB). Here's a CTF cfg which includes all CTF maps in the mapcycle. Paste it in a new file, ctf.cfg, in the baseq3 directory:

```
set sv_hostname "Foofighters CTF"
set sv_maxclients 16
set g_motd "To be or not to be.."
set g_forcerespawn 10
set rconpassword "password"
set g_gametype 4

set m1 "capturelimit 8; map q3ctf1; set nextmap vstr m2"
set m2 "capturelimit 8; map q3ctf2 ; set nextmap vstr m3"
set m3 "capturelimit 8; map q3ctf3 ; set nextmap vstr m4"
set m4 "capturelimit 8; map q3wctf1 ; set nextmap vstr m5"
set m5 "capturelimit 8; map q3wctf2 ; set nextmap vstr m6"
set m6 "capturelimit 8; map q3wctf3 ; set nextmap vstr m1"

vstr m1
```

Start with `./q3ded +exec ctf.cfg`.

5.5. Rocket Arena 3

Rocket Arena 3 is the popular Quake3 version of Rocket Arena. You need the following files:

- ftp://ftp.sunet.se/pube/os/FreeBSD/ports/distfiles/ra315sv.zip
- ftp://ftp.sunet.se/pube/os/FreeBSD/ports/distfiles/ra315cl_linuxmac.zip − required client files (maps).

Create a directory for RA3 and extract the files:

```
$ cd /usr/local/games/quake3
$ mkdir arena
$ unzip ~/ra315sv.zip -d arena
$ unzip ~/ra315cl_linuxmac.zip -d arena
```

Edit `server.cfg` to your liking. Start the server with the following command:

```
$ ./q3ded +set fs_game arena +set sv_pure 0 +bot_enable 0 +set dedicated 2 +set net_port 27960 +e
```

Use `+set dedicated 1` to run a private server or `+set dedicated 2` to run a public. For more info on RA3 configuration options, read `readsrv.txt` in your `arena` directory.

5.6. Alliance

Alliance is a popular CTF/DM mod. It offers three different game styles:

- *Alliance* − Enhanced CTF with offhanded grappling hook (players can use the grapple and shoot their weapons at the same time).
- *Combat* − Resembles the mod Expert Q2. Players spawn with all weapons plus health and ammo regenerate. Slightly different weapons balance, physics and grappling hook style.
- *Instagib* − An interesting (and very entertaining) game type where players spawn with an enhanced rail gun with an infinite ammount of ammo. When you shoot your enemy they are instantly gibbed (killed), so health is not an issue here (also, the rail gun is the only weapon).

All the above game styles can be played as CTF, FFA, Team DM or Tourney. Note that the mod was made with CTF in mind. Anyway, you need the following files:

- http://www.hlrs.de/people/frenzel/actf/alliance30.zip
- http://www.planetquake3.net/download.php?op=viewdownloaddetails=731=Alliance_3.3_Upgrade_from_3.0_
  − 3.0 to 3.3 upgrade.

Extract the files:

```
$ cd /usr/local/games/quake3
$ unzip ~/alliance30.zip
$ unzip ~/alliance30-33.zip
```

Now start the server running your desired gametype. Alliance:

```
$ ./q3ded +set dedicated 2 +set fs_game alliance30 +g_gametype 4 +exec sv_alliance.cfg +map actf1
```

Combat:

```
$ ./q3ded +set dedicated 2 +set fs_game alliance30 +g_gametype 4 +exec sv_combat.cfg +map actf13
```

Instagib:

```
$ ./q3ded +set dedicated 2 +set fs_game alliance30 +g_gametype 4 +exec sv_instagib.cfg +set fs_ba
```

Edit `maplist.cfg` to change the map rotation. Two large Alliance map packs can be downloaded from its homepage.

6. Half−Life

Half−Life is the most popular multiplayer first−person shooter on the Internet as of this version of the HOWTO − mainly thanks to the modification Counter−Strike, hereafter referred to as CS.

6.1. System requirements

The system requirements vary depending on how many players you have on your server.

- CPU − Depends on the number of clients. At least P2 266 for hosting a full game, the more the better. 400MHz recommended.
- RAM − Minimum 64 MB, 128 MB recommended.
- Bandwidth − Also depends on the number of clients. At least 512kbps upstream recommended.

6.2. Installing

You need the HLDS (dedicated server) package to run a Half−Life server:

- ftp://3dgamers.in−span.net/pub/3dgamers2/games/halflife/hlds_l_3110_full.bin

When you've downloaded the above file, create a directory for HLDS and run the downloaded file (specify $HOME/hlds as the directory to install to, if that's what you want):

```
$ mkdir hlds
$ sh hlds_l_3110_full.bin
```

Now we want hlds to see its dynamically linked libraries (*.so files). If you're running bash, which you most probably are, run:

```
$ export LD_LIBRARY_PATH=$HOME/hlds:$LD_LIBRARY_PATH
```

This adds `$HOME/hlds` to the directories ld checks for libraries in.

6.3. Configuring

Now you can start hlds; `./hlds_l` to start a dedicated Half−Life server. It should work with the default settings, so try it:

```
$ hlds_l +maxplayers 12 +map crossfire
```

At the end you should see something like this:

```
WON Auth Server
 . . .
```

If the 'Auth' is missing it's probably because your server is using an incorrect IP address. Type status in the server console and it will spit out some info. If your IP isn't what it should be, quit the server with the exit command and start it again with this command:

```
$ ./hlds_l +ip xx.xx.xx.xx +maxplayers 12 +map crossfire
```

If it doesn't work now, you might be behind a firewall. If that's the case, let's just hope the network admin is friendly − make him open the port(s) you need. 27015 is the default port.

If you want to run a modification instead of regular DM or teamplay, pass the −game modname argument to hlds. Example: `./hlds_l −game tfc +maxplayers 12 +map 2fort`.

The server reads autoexec.cfg, server.cfg, motd.txt and mapcycle.txt. Edit these to your liking. Autoexec.cfg is only executed when you start the server. Server.cfg is executed at every map change. Motd.txt contains the text that players will see when they join your server; a good idea is to put your server name, homepage (server stats page?) and contact info in it. Mapcycle.txt contains the list of maps you want rotated on your server. Type in all the maps you want there, each on its own line, without the ".bsp" extension.

- `sv_aim` − Setting this to 1 causes the crosshairs to turn red when placed over a player.
- `pausable` − Clients are allowed to pause the server.
- `mp_timelimit` − Server changes map when timelimit (in minutes) is reached.
- `mp_fraglimit` − Server changes map when fraglimit is reached.
- `mp_footsteps` − Setting this to 0 turns off footsteps.
- `mp_flashlight` − Clients can use flashlights (useful on dark maps).
- `mp_falldamage` − Realistic fall damage.
- `mp_friendlyfire` − Allows players to kill people on the same team. More realistic, but this can be very problematic without active admins playing on the server, so on public servers it might be a good idea to set this to 0 to avoid lamers killing their own team members.

Min/max rates − The commands `sv_maxrate xxxx` and `sv_minrate xxxx` force clients to use the specified rate. This is useful if you, for example, don't want LPB's stealing bandwidth (which makes HPW's annoyed) − type `sv_maxrate 6000` to do this. If you on the other hand want to run a server for LPB's, set the minimum rate to something around 9000 with `sv_minrate 9000`.

Kicking/Banning − Each player is assigned an userid by the server. Type `users` in the server console to see the players id's. Usually you can kick people simply by typing `kick nick`, but this can be difficult if a client has a long nick or is using strange characters. If this is the case, you can kick clients by their id with `kick #`. If you want to ban a client, preventing them from coming back to the server, find their uniqueid with the `users` command and then ban with `banid minutes uniqueid`, where minutes is the number of minutes you want to ban the player (0 makes the ban permanent), and uniqueid the player's uniqueid. To keep players banned even after restarting the server, type `writeid` before you quit the server, and add `exec banned.cfg` to your server.cfg so that the next time you start your server, it will read the uniqueid's in banned.cfg. You can also kick and ban a client at the same time with `banid uniqueid kick`. Remove a ban with `removeid id`, and don't forget to remove it from server.cfg if it's there too.

RCON − RCON lets clients on your server execute server commands remotely. Add the line `rcon_password "password"` to server.cfg. Clients type rcon_password password in the console and can then execute server commands, like rcon kick, rcon mapchange, etc. This is good if you want to deal with abusive players directly when you're playing on the server, or maybe let a few other trustworthy persons do it. To use rcon when playing on your server, type `rcon_password password` in the in−game console. You can now execute several commands if the password was correct, but remember that you must add rcon before every command, for example `rcon kick player`.

6.4. Counter−Strike

First you need, of course, a working Half−Life server. Download the latest CS package. As of this revision of the HOWTO it's 1.5, but that may have changed when you read this.

- ftp://ftp.gamesdomain.co.uk:21/pub/games/halflife/mods/counterstrike/linux/cs_15_full.tar.gz

Extract it to your HLDS directory:

```
$ tar zxvf cs_15_full.tar.gz −C $HOME/hlds
```

Now run the CS server:

```
$ hlds_l −game cstrike +maxplayers 12 +map de_dust
```

Edit the `/usr/local/games/hlds/cstrike/server.cfg` file to change it to your liking and optimize the server. The file `mapcycle.txt` contains the list of maps to be included in the map cycle, and `motd.txt` contains the MOTD − Message Of The Day.

Maps − By typing mp_timelimit 1 in the console, the server will go to the next map in the cycle. The command changelevel followed by the mapname makes the server change to the specified map. The maps must be in your cstrike/maps directory.

For more information on CS servers, see http://server.counter−strike.net.

7. Unreal Tournament

Unreal Tournament is a very addictive first person shooter which is designed to be run in one of several modes; DeathMatch, Capture the Flag, Last Man Standing and Assault are the most popular. It features bots, a bevy of weapons and plenty of community support. It was originally ported as an in−house development effort at Epic but later releases are handled by Loki Software. Current version as of this writing is 436.

7.1. System requirements

As always, the system requirements will vary slightly depending on the number of players on the server. Here are the basics:

- Pentium 266 (For more than 8 users, a faster server is recommended)
- At least 32 MB RAM (For a VERY dedicated machine. The more maps, skins, mutators, bots, and players that are online, the more RAM the server will use.) 64 MB is highly recommended.
- Linux kernel 2.2.14 or later, Glibc 2.1 or later.
- The retail Unreal Tournament CD (Just the first one is necessary).
- At least 500 MB disk space. Maps, skins, and mods will require a bit more.
- Enough bandwidth if you're going to run an Internet server. Epic recommends about 20kbps per player.

7.2. Installing

First, download the latest ut−install file from Loki's website. As of this version of the HOWTO, that is 436.

- http://www.lokigames.com/products/ut/updates.php3

Mount the Unreal Tournament CD (Usually /cdrom or /mnt/cdrom), and run the ut−install*.run file. It will verify the archive, check the CD for the appropriate files, and begin installing. It will ask you where you want to install UT to, as well as a few other questions. It is actually fairly painless.

Once you have it installed, you will want to change ownership of the files to the user ID that they will be running under (See Security and permissions).

7.3. Configuring

Now it's time to write a config file. Unreal Tournament uses an INI−style config file format which is exactly like Windows INI files. This config file will contain ALL the variables the server will use, including game types, defaults for those types, number of players, bots, etc. The cfg's should be kept in the System/ directory (either `$UTROOT/System/` or `~/.loki/ut/System/`). If you have an UnrealTournament.ini and/or User.ini in that System/ directory, it will be loaded and can override settings in the server's cfg file. This is a sanitized copy of my server's cfg file (called ucc.ini):

```
----[snip]-----
[URL]
Protocol=unreal
ProtocolDescription=Unreal Protocol
Name=Player
Map=Index.unr
```

```
LocalMap=DM−Deck16][.unr
Host=
Portal=
MapExt=unr
SaveExt=usa
Port=7777
Class=Botpack.TMale1

[FirstRun]
FirstRun=428

[PackageRemap]
UnrealShare=UnrealI

[Engine.Engine]
GameRenderDevice=NullDrv.NullRenderDevice
WindowedRenderDevice=NullDrv.NullRenderDevice
RenderDevice=NullDrv.NullRenderDevice
AudioDevice=NullDrv.NullRenderDevice
NetworkDevice=IpDrv.TcpNetDriver
DemoRecordingDevice=Engine.DemoRecDriver
Console=UTMenu.UTConsole
Language=int
GameEngine=Engine.GameEngine
EditorEngine=Editor.EditorEngine
DefaultGame=Botpack.DeathMatchPlus
DefaultServerGame=Botpack.DeathMatchPlus
ViewportManager=SDLDrv.SDLClient
Render=Render.Render
Input=Engine.Input
Canvas=Engine.Canvas

[Core.System]
PurgeCacheDays=30
SavePath=../Save
CachePath=../Cache
CacheExt=.uxx
Paths=../System/*.u
Paths=/path/to/ut/Maps/*.unr
Paths=/path/to/ut/Textures/*.utx
Paths=/path/to/ut/Sounds/*.uax
Paths=/path/to/ut/Music/*.umx
Suppress=DevLoad
Suppress=DevSave
Suppress=DevNetTraffic
Suppress=DevGarbage
Suppress=DevKill
Suppress=DevReplace
Suppress=DevSound
Suppress=DevCompile
Suppress=DevBind
Suppress=DevBsp

[Engine.GameEngine]
CacheSizeMegs=4
UseSound=False
ServerActors=IpDrv.UdpBeacon
ServerActors=IpServer.UdpServerQuery
;ServerActors=IpServer.UdpServerUplink MasterServerAddress=unreal.epicgames.com MasterServerPort=
;ServerActors=IpServer.UdpServerUplink MasterServerAddress=master0.gamespy.com MasterServerPort=2
;ServerActors=IpServer.UdpServerUplink MasterServerAddress=master.mplayer.com MasterServerPort=27
ServerActors=UWeb.WebServer
```

```
ServerPackages=SoldierSkins
ServerPackages=CommandoSkins
ServerPackages=FCommandoSkins
ServerPackages=SGirlSkins
ServerPackages=BossSkins
ServerPackages=Botpack

[Engine.Player]
ConfiguredInternetSpeed=20000
ConfiguredLanSpeed=20000

[IpDrv.TcpNetDriver]
AllowDownloads=True
ConnectionTimeout=15.0
InitialConnectTimeout=300.0
AckTimeout=1.0
KeepAliveTime=0.2
MaxClientRate=20000
SimLatency=0
RelevantTimeout=5.0
SpawnPrioritySeconds=1.0
ServerTravelPause=4.0
NetServerMaxTickRate=20
LanServerMaxTickRate=35
DownloadManagers=IpDrv.HTTPDownload
DownloadManagers=Engine.ChannelDownload

[Engine.DemoRecDriver]
DemoSpectatorClass=Botpack.CHSpectator
MaxClientRate=25000
ConnectionTimeout=15.0
InitialConnectTimeout=500.0
AckTimeout=1.0
KeepAliveTime=1.0
SimLatency=0
RelevantTimeout=5.0
SpawnPrioritySeconds=1.0
ServerTravelPause=4.0
NetServerMaxTickRate=60
LanServerMaxTickRate=60

[Engine.GameReplicationInfo]
ServerName=Generic UT Server
ShortName=UT Server
AdminName=Lamer
AdminEmail=root@localhost
Region=0
MOTDLine1=
MOTDLine2=
MOTDLine3=
MOTDLine4=

[IpDrv.TcpipConnection]
SimPacketLoss=0
SimLatency=0

[IpServer.UdpServerUplink]
DoUpLink=False
UpdateMinutes=1
MasterServerPort=27900

[IpServer.UdpServerQuery]
```

```
GameName=ut

[IpDrv.UdpBeacon]
DoBeacon=True
BeaconTime=0.50
BeaconTimeout=5.0
BeaconProduct=ut

[UMenu.UnrealConsole]
RootWindow=UMenu.UMenuRootWindow
UWindowKey=IK_Esc
ShowDesktop=False

[UMenu.UMenuMenuBar]
ShowHelp=True
GameUMenuDefault=UTMenu.UTGameMenu
MultiplayerUMenuDefault=UTMenu.UTMultiplayerMenu
OptionsUMenuDefault=UTMenu.UTOptionsMenu
ModMenuClass=UMenu.UMenuModMenu

[Botpack.ChallengeBotInfo]
Difficulty=1

[Botpack.DeathMatchPlus]
bNoviceMode=True
bHardCoreMode=False
bUseTranslocator=True
bCoopWeaponMode=False
MinPlayers=8
AirControl=0.350000
bChangeLevels=True
bMegaSpeed=False
bAltScoring=False
bTournament=False
NetWait=10
RestartWait=15
InitialBots=10
FragLimit=30
TimeLimit=0
bMultiWeaponStay=False
bForceRespawn=False
MaxCommanders=0
bNoMonsters=False
bHumansOnly=False
bClassicDeathMessages=False

[Botpack.CTFGame]
bUseTranslocator=True
bCoopWeaponMode=True
GoalTeamScore=3.000000
bNoTeamChanges=False
FriendlyFireScale=0.000000
MaxTeams=2
MaxTeamSize=16
FragLimit=0
TimeLimit=0
bMultiWeaponStay=True
bForceRespawn=False
MaxCommanders=0
bNoMonsters=False
bHumansOnly=True
bClassicDeathMessages=False
```

```

[Botpack.Domination]
bDumbDown=True
bUseTranslocator=True
bCoopWeaponMode=True
GoalTeamScore=100.000000
bNoTeamChanges=False
FriendlyFireScale=0.000000
MaxTeams=2
MaxTeamSize=16
FragLimit=30
TimeLimit=0
bMultiWeaponStay=True
bForceRespawn=False
MaxCommanders=0
bNoMonsters=False
bHumansOnly=False
bClassicDeathMessages=False

[Botpack.Assault]
bUseTranslocator=False
bCoopWeaponMode=True
Defenses=3
SavedTime=0.000000
NumDefenses=0
CurrentDefender=0
bDefenseSet=False
bTiePartOne=False
GameCode=
Part=1
bNoTeamChanges=False
FriendlyFireScale=0.000000
MaxTeams=2
GoalTeamScore=0.000000
MaxTeamSize=16
FragLimit=0
TimeLimit=7
bMultiWeaponStay=False
bForceRespawn=False
MaxCommanders=2
bNoMonsters=False
bHumansOnly=False
bClassicDeathMessages=False

[Botpack.TeamGamePlus]
bBalanceTeams=True
GoalTeamScore=30
bPlayersBalanceTeams=True

[Engine.GameInfo]
bLowGore=False
bVeryLowGore=False
bMuteSpectators=False
bNoCheating=True
bAllowFOV=False
AutoAim=0.930000
GameSpeed=1.000000
MaxSpectators=2
AdminPassword=
GamePassword=
MaxPlayers=16
IPPolicies[0]=ACCEPT,*
```

```
IPPolicies[1]=
IPPolicies[2]=
IPPolicies[3]=
IPPolicies[4]=
IPPolicies[5]=
IPPolicies[6]=
IPPolicies[7]=
IPPolicies[8]=
IPPolicies[9]=
IPPolicies[10]=
IPPolicies[11]=
IPPolicies[12]=
IPPolicies[13]=
IPPolicies[14]=
IPPolicies[15]=
IPPolicies[16]=
IPPolicies[17]=
IPPolicies[18]=
IPPolicies[19]=
IPPolicies[20]=
IPPolicies[21]=
IPPolicies[22]=
IPPolicies[23]=
IPPolicies[24]=
IPPolicies[25]=
IPPolicies[26]=
IPPolicies[27]=
IPPolicies[28]=
IPPolicies[29]=
IPPolicies[30]=
IPPolicies[31]=
IPPolicies[32]=
IPPolicies[33]=
IPPolicies[34]=
IPPolicies[35]=
IPPolicies[36]=
IPPolicies[37]=
IPPolicies[38]=
IPPolicies[39]=
IPPolicies[40]=
IPPolicies[41]=
IPPolicies[42]=
IPPolicies[43]=
IPPolicies[44]=
IPPolicies[45]=
IPPolicies[46]=
IPPolicies[47]=
IPPolicies[48]=
IPPolicies[49]=
ServerLogName=server.log
bLocalLog=True
bWorldLog=True
bBatchLocal=False
DemoBuild=0
DemoHasTuts=0
bExternalBatcher=False
bNoMonsters=False
bHumansOnly=False
bCoopWeaponMode=False
bClassicDeathMessages=False

[UnrealShare.UnrealGameOptionsMenu]
```

```
bCanModifyGore=True

[UWeb.WebServer]
Applications[0]=UTServerAdmin.UTServerAdmin
ApplicationPaths[0]=/ServerAdmin
Applications[1]=UTServerAdmin.UTImageServer
ApplicationPaths[1]=/images
DefaultApplication=0
bEnabled=True
ListenPort=5080
MaxConnections=30

[UTServerAdmin.UTServerAdmin]
AdminUsername=utadmin
AdminPassword=utpasswd
MenuPage=menu
RootPage=root
CurrentPage=current
CurrentMenuPage=current_menu
CurrentIndexPage=current_index
CurrentPlayersPage=current_players
CurrentGamePage=current_game
CurrentConsolePage=current_console
CurrentConsoleLogPage=current_console_log
CurrentConsoleSendPage=current_console_send
DefaultSendText=say
CurrentMutatorsPage=current_mutators
CurrentRestartPage=current_restart
DefaultsPage=defaults
DefaultsMenuPage=defaults_menu
DefaultsMapsPage=defaults_maps
DefaultsRulesPage=defaults_rules
DefaultsSettingsPage=defaults_settings
DefaultsBotsPage=defaults_bots
DefaultsServerPage=defaults_server
DefaultsIPPolicyPage=defaults_ippolicy
DefaultsRestartPage=defaults_restart
MessageUHTM=message.uhtm
DefaultBG=#aaaaaa
HighlightedBG=#ffffff
AdminRealm=UT Remote Admin Server

[IpDrv.HTTPDownLoad]
UseCompression=True

[Engine.StatLog]
LocalBatcherURL=../NetGamesUSA.com/ngStats/ngStatsUT
LocalBatcherParams=
LocalStatsURL=../NetGamesUSA.com/ngStats/html/ngStats_Main.html
WorldBatcherURL=../NetGamesUSA.com/ngWorldStats/bin/ngWorldStats
WorldBatcherParams=−d ../NetGamesUSA.com/ngWorldStats/logs −g UT
WorldStatsURL=http://www.netgamesusa.com
LocalLogDir=../Logs
WorldLogDir=../NetGamesUSA.com/ngWorldStats/logs
bWorldBatcherError=False

[Botpack.TrainingDM]
FragLimit=3
TimeLimit=0
bMultiWeaponStay=True
bForceRespawn=False
bUseTranslocator=False
```

```
MaxCommanders=0
bNoMonsters=False
bHumansOnly=False
bCoopWeaponMode=False
bClassicDeathMessages=False

[Botpack.TrainingDOM]
bDumbDown=True
bNoTeamChanges=False
FriendlyFireScale=0.000000
MaxTeams=2
GoalTeamScore=100.000000
MaxTeamSize=16
FragLimit=0
TimeLimit=0
bMultiWeaponStay=True
bForceRespawn=False
bUseTranslocator=True
MaxCommanders=0
bNoMonsters=False
bHumansOnly=False
bCoopWeaponMode=True
bClassicDeathMessages=False

[UTMenu.UTServerSetupPage]
bLanPlay=True

[UTMenu.UTStartGameCW]
Map=DM−Deck16][.unr
GameType=BotPack.DeathMatchPlus
MutatorList=
bKeepMutators=False

[Botpack.TDMmaplist]
Maps[0]=DM−Liandri.unr
Maps[1]=DM−Codex.unr
Maps[2]=DM−Grinder.unr
Maps[3]=DM−Pressure.unr
Maps[4]=DM−HyperBlast.unr
Maps[5]=DM−Peak.unr
Maps[6]=DM−KGalleon.unr
Maps[7]=DM−Turbine.unr
Maps[8]=DM−StalwartXL.unr
Maps[9]=DM−Curse][.unr
Maps[10]=DM−Deck16][.unr
Maps[11]=DM−Phobos.unr
Maps[12]=
Maps[13]=
Maps[14]=
Maps[15]=
Maps[16]=
Maps[17]=
Maps[18]=
Maps[19]=
Maps[20]=
Maps[21]=
Maps[22]=
Maps[23]=
Maps[24]=
Maps[25]=
Maps[26]=
Maps[27]=
Maps[28]=
Maps[29]=
Maps[30]=
Maps[31]=
MapNum=0

[Botpack.TrainingCTF]
bNoTeamChanges=False
FriendlyFireScale=0.000000
MaxTeams=2
GoalTeamScore=3.000000
MaxTeamSize=16
FragLimit=0
TimeLimit=0
bMultiWeaponStay=True
bForceRespawn=False
bUseTranslocator=True
MaxCommanders=0
bNoMonsters=False
bHumansOnly=True
bCoopWeaponMode=True
bClassicDeathMessages=False

[UTMenu.UTMenuBotmatchCW]
Map=DM−Gothic.unr
GameType=BotPack.DeathMatchPlus
MutatorList=
bKeepMutators=False

[Botpack.ASMapList]
Maps[0]=AS−Hispeed.unr
Maps[1]=AS−Frigate.unr
Maps[2]=AS−Rook.unr
Maps[3]=AS−Mazon.unr
Maps[4]=AS−OceanFloor.unr
Maps[5]=AS−Overlord.unr
Maps[6]=AS−Guardia.unr
Maps[7]=
Maps[8]=
Maps[9]=
Maps[10]=
Maps[11]=
Maps[12]=
Maps[13]=
Maps[14]=
Maps[15]=
Maps[16]=
Maps[17]=
Maps[18]=
Maps[19]=
Maps[20]=
Maps[21]=
Maps[22]=
Maps[23]=
Maps[24]=
Maps[25]=
Maps[26]=
Maps[27]=
Maps[28]=
Maps[29]=
Maps[30]=
Maps[31]=
MapNum=0

-----[snip]-----
```

A few things of note here.

The following values should be changed before the server is launched for the first time:

```
[Core.System]
Paths=../System/*.u
Paths=/path/to/ut/Maps/*.unr
Paths=/path/to/ut/Textures/*.utx
Paths=/path/to/ut/Sounds/*.uax
Paths=/path/to/ut/Music/*.umx
```

Should all be set to where Unreal Tournament is installed.

```
[Engine.GameEngine]
;ServerActors=IpServer.UdpServerUplink
```

Should be uncommented if you wish to link your Unreal Tournament server into an existing Unreal Tournament league and have it accessible over the Internet.

```
[IpServer.UdpServerUplink]
DoUpLink=False
```

Additionally, if you do want to link your UT server with an Internet league, you will need to allow UpLinking.

```
[Engine.Player]
ConfiguredInternetSpeed=20000
```

This should be set to the allocated bandwidth (in bytes per second) for each player. Leaving it at the recommended 20000 is good.

```
[Engine.GameReplicationInfo]
ServerName=Generic UT Server
ShortName=UT Server
AdminName=Lamer
AdminEmail=root@localhost
Region=0
MOTDLine1=
MOTDLine2=
MOTDLine3=
MOTDLine4=
```

All of that should be changed.

```
[Engine.GameInfo]
AdminPassword=
MaxPlayers=16
```

An admin password should be set, and the Max Players should be set to a sane value (See ConfiguredInternetSpeed).

```
[UWeb.WebServer]
bEnabled=True
ListenPort=5080
```

One of the most interesting aspects of running a UT server is the web−server interface for admins. By setting bEnabled to True and specifying a ListenPort, you can administer an UnrealTournament server via a normal Web Browser.

```
[UTServerAdmin.UTServerAdmin]
AdminUsername=utadmin
AdminPassword=utpasswd
AdminRealm=UT Remote Admin Server
```

If you do decide to run the UT Webserver interface (Recommended), the above values should be set. AdminRealm is less important, but will be used by your browser to know when to ask for the administrator username and password.

All other values should be checked and seasoned to taste.

7.4. Starting the server

To start the Unreal Tournament server, you will need to use the command "ucc server" and specify a few things on the command line. These being:

- Mapname − Name of the map to start from the Maps/ directory, minus the .unr extension.
- Gametype − One of Botpack.DeathMatchPlus, Botpack.Domination, Botpack.CTFGame.
- INI − The config file to use to start the server.
- LOG − the log file the server should use.
- Port − (Optional) the port the UT server should use.

Most of this is specified in the config file, but the mapname and Gametype are required.

Here is an example:

```
$ ucc server "DM−Turbine?game=Botpack.DeathMatchPlus" ini=ucc.ini log=ucc.log
```

I quoted the mapname and gametype because of the presence of the ? (Linux shells use that as a wild card) and because a lot of Unreal Tournament maps have odd characters in their name (DM−Deck16][anyone?). I recommend they always be quoted.

Once the server starts, it will load the settings specified in the System/ucc.ini file and load the game.

I also recommend the server be started with the nohup command and placed in the background. All the configuration will take place in the web interface.

7.5. Administrating the server

Administering the server is as easy as browsing a website. When you started the server, you should have seen something like this toward the end:

```
Spawning: IpDrv.UdpBeacon
Spawning: IpServer.UdpServerQuery
Spawning: UWeb.WebServer
Bound to UWeb.so
```

Notice the UWeb.Webserver and UWeb.so sections? Those are the webserver, and since it didn't give any errors, we know it's running.

Now, in the configuration section, you told it what port to listen on:

```
[UWeb.WebServer]
bEnabled=True
ListenPort=5080
```

So, to connect to the administration server, you would need to browse to
`http://utserver:5080/ServerAdmin`

The first thing that will happen is that your browser will ask you for authentication. Back in the configuration file, we specified that:

```
[UTServerAdmin.UTServerAdmin]
AdminUsername=utadmin
AdminPassword=utpasswd
AdminRealm=UT Remote Admin Server
```

So in this case, you would sign in as utadmin with a password of utpasswd (You had better change these before you actually start the server).

Once you are authenticated, you are presented with an interface that allows you to specify the game type, map, number of bots and connections as well as kick and ban users. It's all very point−n−click.