文件上传服务器使用手册

一、简介

本文件上传服务器简称 HYFileServer，是完全遵照 W3C 的 HTTP 协议开发的专用 Web 文件上传服务器，支持浏览器通过 multipart/form-data 协议上传文件，服务器的使用方法与任何其它 Web 文件上传的服务器使用方法一样，一般来说，如果你现有的浏览器端上传模块能够上传到标准的 Web 服务器上，则无需修改即能向 HYFileServer 服务器上传文件，但是相对于其它 Web 服务器来说是有区别的：

1）HYFileServer 是针对于 HTTP 文件上传的专门实现与优化，因此性能高。

2）HYFileServer 采用即时写入方式，不像其它 WEB 上传方式需要缓存，例如 PHP、JAVA 的上传是先由服务器缓存为临时文件，或者服务器将上传数据缓存到内存中后，再由脚本调用相关的上传文件处理函数来移动临时文件来保存文件数据；由于 PHP、JAVA 等处理文件上传需要分两步，对于大文件与超大文件来说，再次移动文件也是比较耗时间与系统资源的。 即时写入方式是服务器根据上传配置，实时将接收到的上传数据保存到目标位置，无需再次移动文件。

3）HYFileServer 采用异步 I/O 架构设计，具有高性能 I/O 能力，尤其适用于传输大文件与超大文件。

4）采用高效内存分配技术确保在运行过程中内存开销最小化。

5）完全采用标准协议，支持任意现代 PC 浏览器与移动终端浏览器上传文件。

6）支持超大文件上传，基于 PHP、JAVA 等技术的 WEB 服务器天生是对超大文件上传无法支持，无论在服务器端如何配置，上传文件很难超过 2GB 的瓶颈，HYFileServer 则在设计时就考虑了大文件尺寸的支持，因此，理论上对上传文件大小无限制。

二、安装与使用

2.1 安装

将程序包展开后，出现如下目录结构：

1）其中 HYFileServer.exe 文件是上传服务器模块，HYFileServer.exe 是已经将所有依赖的 DLL 模块全部合并以后的结果，因此无须担心找不到依赖的动态库。
2）其中 conf 目录为配置文件目录，该目录下的 config.xml 为服务器用到的唯一配置文件。
3）其中 html 目录为网页目录，安装时无需关注。
4）其中 log 目录为服务器产生的日志目录，安装时无须关注。

打开系统命令行窗口（在 Windows 平台下执行 cmd 即可），然后进入到 HYFileServer.exe 文件所在的目录：

1）安装为系统服务

>HYFileServer.exe –i

命令“**HYFileServer.exe –i**”将 HYFileServer.exe 安装为一个开机自动执行的系统服务。

2）启动文件上传服务

打开系统的“服务管理器”（Windows 下如何打开服务管理器请查看相关资料），找到如下图所示服务：

HYFileServer.exe 在服务管理器中的显示名称为“Hyper Http Upload Service”，将“Hyper Http Upload Service”服务条目选中，然后点击系统服务管理器的上方工具栏的“启动服务”按钮，即可启动服务。由于该服务已经安装为系统的自动启动服务，下次重启计算机时无需再手工启动服务了。

3）将已经安装的系统服务删除

如果 HYFileServer.exe 安装后不再使用，请执行如下命令：
>HYFileServer.exe –u
以上命令将卸载已经安装的文件上传服务。

2.2 配置文件

以下是上传服务器的一个完整的配置文件，配置文件是 XML 格式的文本文件，用任意文本编辑器即可打开进行编辑。

```
<?xml version="1.0" encoding="UTF-8"?>
<hywebfileserver version="1.0">
    <serverconfig
       port="8080"
       root="html"
       tempdir="temp"
       logdir="log"
    />
   <vdirectory>
       <location name="vrview" path="C:/wamp64/htdocs.livechat/www/vrview" />
       <location name="uploadtest" path="C:/wamp64/htdocs.livechat/www/uploadtest" />
   </vdirectory>
   <upload dir=" e:/webroot/uploads" key="123456" dirformat="YMD" baseurl="/uploads">
       <item id="audio" dir="e:/webroot/uploads/audio" key="123456" baseurl="/uploads/audio"/>
       <item id="image" dir="e:/webroot/uploads/images" key="123456" baseurl="/uploads/images" />
       <item id="video" dir="e:/webroot/uploads/videos" key="123456" baseurl="/uploads/video" />
       <item id="document" dir="e:/webroot/uploads/document" key="123456" baseurl="/uploads/doc" />
   </upload>
</hywebfileserver>
```

图 2-1

HYFileServer 是一个上传专用的 Web 服务器，没有过多的功能，因此使用配置非常简单。在这里对这个配置文件进行说明。

简单浏览一下配置文件，即可明白配置文件的基本含义。配置文件必须具备 **hywebfileserver** 这个主 XML 节点，否则，文件服务器无法读取配置文件。

2.1.1 服务器基本配置

任何一个网络服务器程序均需要配置网络端口，HYFileServer 服务器也不例外，基本配置由节点 serverconfig 节点来确定，该节点配置项有 4 项，当前也只有 4 项，下面一一阐述：

port：设置服务器程序使用的 TCP 端口号，取值在 1 到 65535 之间，如果当前服务器主机没有安装过其它 HTTP 服务器，则建议使用 80 端口，否则如果 8080 端口没有使用，建议使用 8080 端口，使用其他任何端口号均可，但务必不要与其它程序的正在使用的端口号冲突。上传服务器不需要配置 IP 地址，上传服务器启动后会自动绑定本机的所有 IP 地址，即通过本机的任意网卡都能访问，因此本机具备多网卡时不用重复配置即可用多网卡进行上传。

root：文档根目录，上传服务器内置了 HTTP 传输服务功能，能够对静态文件提供 HTTP 下载服务，这里设置的是根目录的位置，在这里设置好后，就可以通过 HTTP 协议访问本服务器上的文件与网页了，默认为 html，是位于上传服务器主程序所在的目录下的一个 html 的子目录，放在这个目录下的文件可以通过 http 协议来访问，例如：输入 htttp://<IP>:<port>/index.html 即可访问该目录下的 index.html 文件。

在 Windows 下如果不带盘符，例如 C:、D:，则此处的目录为相对于主程序目录的目录地址，否则为绝对目录地址，可以将根目录配置未任何可以访问的目录地址。在 Linux 下，如果不是以“/”根路径符号开始的目录是相对目录，否则为绝对目录。

注意：如果没有特别说明，<IP>:<port>代表本上传服务器的网络地址，其中<IP>代表服务器的 IP 地址，:<port>代表端口号，如果:<port>为:80，则端口号可以省略。

tempdir：设置服务器工作时产生的临时文件的保存目录，默认为 temp，用默认的即可。

logdir：设置服务器运行时的日志文件保存目录，默认为 log，用默认的即可。

注意：以上目录设置，**root** 处设置的目录需要有读取权限，**tempdir** 与 **logdir** 处设置的目录需要有写入权限。

2.1.2 上传目录配置

文件上传的过程是由浏览器将文件向文件服务器传送，在传送过程中上传服务器将远程的文件数据保存为服务器的本地文件，在文件传送完毕后上传服务器再向浏览器反馈上传的信息，然后整个传送过程结束。上传服务器需要知道将文件保存到服务器主机的哪个目录以及如何将文件进行分类存储，这一节就对这个主题进行说明。这一节还涉及到上传服务器如何命名文件以及如何向终端返回上传信息的问题。

上传配置是由节点 upload 来确定，该节点有 4 个配置属性项，下面意义阐述：

dir: 上传文件保存的物理路径，必须是绝对路径，在路径后面不要带“ /”(Linux)或者“\”(Windows)目录分割符。上传目录必须保证有写入权限，上传目录可以是任何位置，由于上传的文件一般需要被其它 Web 服务器（例如 Apache、Tomcat）访问，因此建议将上传的路径设置为其它 Web 服务器的虚拟目录。

注意：在 Windows 平台下目录分隔符“\”一定要用“/”替换。

key: 文件名加密的一个密钥值，可以为任何字符串，上传服务器将这个值与传送过来的文件名值进行 MD5 加密后产生新的目标文件名称。

dirformat: 目录命名格式，由于上传服务器管理的上传文件一般比较多，很多的时候都需要将文件按照上传时间分别存储在上传目录的子目录里，要实现这个功能，就需要配置这个值。上传服务器支持对目录进行自行灵活命名，命名格式支持 Y、M、D、H、N 与 S 共 6 个宏，他们的含义分别为：

Y：当前 4 为年号，例如 2007、2017 等

M：当前 2 位月份，从 01、02 直到 12
D：当前 2 位日子，每月的日号，从 01 到 31 日
H：当前小时数，从 01 到 24
N：当前分钟数，从 01 到 59
S：当前分钟的秒数，从 01 到 59。

如果确实需要在目录中出现这几个字母，则请在这几个字母前插入“%”。

例如，假设当前日期为 2017 年 07 月 01 日 20:15:23，日期命名如下：
TMD：将被替换为：20170701 ，
T-M-D 将被替换为：2017-07-01，
DMT：将被提行为：01072017 ，
D-T-M 将被替换为：01-2017-07
VIDEOTMD：将被替换为：VI07EO20170701，
VI%DEOTMD：将被替换为：VIDEO20170701
一般来说，我们命名子目录时只要精确到每日即可，因此用“TMD”作为命名格式即可。如果不需要根据时间来创建子目录，请将该项删除。如果该项不存在，则上传服务器不会再根据时间创建子目录。

baseurl：用于访问上传后的文件的 URL 的基地址，这个基地址将作为前缀与生成的文件名合起来产生一个 HTTP 协议访问的 URL，举例说明如下：
假设当前服务器主机的 IP 地址为 192.168.1.100，服务器上除了运行有上传服务器之外，还运行了 Apache 服务器，Apache 的文档根目录为 c:\webroot，将上传服务器的 **dir** 参数配置为 c:\webroot\uploads 目录，Apache 的端口为 80，则此处的基地址地址就是/uploads，如果上传了一个文件，上传服务器将它保存在 c:\webroot\uploads\20170701\aaaa.mp4 位置，则上传服务器会根据基地址/uploads 产生一个用于访问这个文件的 URL，为：/uploads/20170701/aaaa.mp4。
如果使用 WEB 服务器与上传服务器安装在同一台主机上，基地址可以省略 HTTP 主机前缀，如果上传服务器与 WEB 服务器安装在不同的主机上，它们之间共享存储设备，则配置 **baseurl** 时，HTTP 主机前缀不能省略。举个例子如下：
假设上传服务器安装在 192.168.1.100 主机上，WEB 服务器安装在 192.168.1.120 主机上，它们通过 NAS 共享存储设备，将上传目录配置为 c:\webroot\uploads，而这个位置相对于 192.168.1.120 主机来说，是该主机上的 Apache 服务器的文档目录下的 uploads 目录，那么，**baseurl** 应该配置为完整地址前缀，如下：
http:// 192.168.1.120/uploads。
在上传完毕后，上传服务器产生的 URL 如下：
http:// 192.168.1.120/uploads/20170701/aaaa.mp4

通常，我们上传的文件需要根据文件分类不同上传到不同的目录下以便于管理。HYFileServer 文件服务器实现了这个功能。
在上传目录配置节点 **upload** 里可以添加子节点，每个子节点代表一类文件的保存目录。子节点的元素标签为 item，每个 item 代表一种上传的文件分类，可以自由添加任意数量的 item。
节点 item 的属性设置说明如下：

id：节点标识，其值自行定义，用于标识这个文件上传类的 id，同时用于上传的目录后缀；
dir：该分类文件的存储目录，与 upload 节点的 dir 意义相同；

baseurl：访问该分类文件的 http 基地址，与 upload 节点的 baseurl 意义相同；
key：文件名加密的一个密钥值，与 upload 节点的 key 意义相同；

通常情况下需要将视频、音频、图像与文档放置在不同的目录下，因此可以在节点 **upload** 里添加诸如 id 为 video、audio、image 与 document 等值的节点。

上传服务器的基本上传地址为 http:// <IP>:<port>/upload，其中 upload 是 HYFileServer 内置的路径，**不可更改**，如果要向分类目录下上传文件，将基本上传地址与分类目录的 id 组合起来即可。如 2.2 示范配置文件所示，如果要上传到 e:/webroot/uploads/videos 目录下，用的上传地址如下：
http:// <IP>:<port>/upload/video
其中 video 为 item 的 id 值。

2.1.3 虚拟目录配置

HYFileServer 文件服务器内置了静态文件下载功能，可以作为大文件与超大文件的 HTTP 下载传输之用，也可以作为静态网页访问之用。由于 HYFileServer 服务器采用异步 I/O 技术，因此它作为下载服务器具有比较高的性能，内建同时支持多用户并发下载大文件，在 Windows 平台上支持最高不超过 1000 个并发下载连接。

虚拟目录节点由 vdirectory 来设置，vdirectory 节点没有附加属性，vdirectory 节点的目录由子节点 location 来管理，每个 vdirectory 节点最多添加 4 个 location 节点，也就是说，HYFileServer 可以配置最多不超过 4 个虚拟目录。

Location 节点的属性如下：
name：虚拟目录名称，可以自由定义；
path：虚拟目录对应的物理目录，必须是服务器硬盘上已经存在的真实目录。
例如：假设服务器硬盘上存在目录 C:/Resources/Videos，我们需要将这个目录下的文件提供 HTTP 下载服务（也可以用于 HTML5 播放服务），只需配置一个指向该目录的虚拟目录，在这里，我们将虚拟目录名称命名为“videos”，则在 vdirectory 节点下添加一条记录如下：

```
<vdirectory>
    <location name="videos" path="C:/Resources/Videos" />
</vdirectory>
```

假设 目录 C:/Resources/Videos 下存在 xxxx.mp4 媒体文件，
我们通过 http:// <IP>:<port>/videos/xxxx.mp4 这个 URL 即可访问 C:/Resources/Videos 目录下的媒体文件了。
HYFileServer 可以配置一个 root 目录与 4 个虚拟目录，用于静态文件访问。

三、开发说明

这一节适合于开发人员阅读。
采用 HYFileServer 文件上传服务器进行文件上传的浏览器端开发与采用其它任何 HTTP 服务器进行上

传的开发技术一模一样，只要浏览器端支持 multipart/form-data 协议即可进行文件上传。

由于 HYFileServer 服务器作为独立的服务器与 WEB 应用系统采用的 HTTP 服务器不是同一个进程，因此，HYFileServer 服务器与 WEB 应用系统采用的 HTTP 服务器具备不同的 TCP 端口，而且也许可能具备不同的 IP 地址，因此整合开发时需要做出正确的配置，需要将上传地址指向 HYFileServer 服务器的地址。

一般 WEB 应用系统文件上传的的流程是：

1）选择文件

2）上传到服务器，并获得反馈信息

3）将文件反馈信息与其它数据一并上传到 WEB 应用服务器并保存到数据库里。

将文件上传与信息提交合并在一个 POST 请求里弊端多多，一个是加重服务器的分析负担导致不稳定，另外很容易导致提交失败。因此采用 HYFileServer 上传服务器的建议模式就是先上传文件，在上传文件成功并获得反馈信息后再提交其它数据。

3.1 返回信息说明

HYFileServer 服务器采用 JSON 来返回信息，返回的信息包括浏览器端的原始文件名、服务器的物理文件名、映射的访问 URL 与文件尺寸，以下是一个成功提交返回的 JSON 数据：

```
{
 "result":"success",
 "msg":"file upload success!",
 "files": [
  {
  "file": "Live Channel 120160415121006_1.mp4",
  "url": "/Uploads/videos/20170724/cdb4dd5c84e97ff547e16ee8c0ff865b.mp4",
  "path":"G:/Uploads/videos/20170724/cdb4dd5c84e97ff547e16ee8c0ff865b.mp4",
  "filesize":"121562994"
  }
 ]
}
```

如果上传成功，result 字段的值为 success。

返回的文件数据放置在一个 files 数组里，由于 HYFileServer 服务器支持一个 POST 请求里发送多个文件，因此 files 也许会包含多个文件对象。

用浏览器自带的 JSON 分析器（老的浏览器不带 JSON 分析器，但所有浏览器支持 eval 函数，因此可以用 eval 函数来分析 JSON）进行分析。如果采用乐 jQuery，可以用 jQuery 的 parseJSON 方法来分析。

由于返回的信息已经十分明了，因此不再一一解释每个成员的含义。

一个上传失败的返回信息可能如下：

```
{
 "result": "error",
 "msg": "server internal error: No such file or directory"
}
```

注意：如果上传失败，服务器将返回错误信息，"result"字段值未"error"，通常在上传小文件出错时能够返回错误 JSON 信息，但如果上传一个大文件，如果出错发生在浏览器尚未将文件数据传送完毕，这时由于 HYFileServer 服务器已经将连接断开，浏览将会报错，错误为连接重置，浏览器端无法获得 JSON 反馈信息的，这时仍旧可以在浏览器端捕获这个错误。

上传文件失败的原因可能是以下：

1）服务器硬盘空间不够

2）服务器没有目录写入权限

3）服务器上的物理路径根本不存在

4）服务器硬盘读写失败。

由于即使不能获得 HYFileServer 返回的失败信息，我们也能判断文件上传失败，因此可以从失败信息来查找错误。注意：出现以上几种情况时，大多数 WEB 服务器的行为与 HYFileServer 服务器一样，都会提前关闭连接，因此请注意从以上几个方面排查问题。

3.2 文件上传调用说明

文件上传过程也许需要一个漫长的过程，文件上传主流的界面设计均采用无刷新技术，也就是说，文件上传完毕后不会改变远操作界面，这样能够大大提高用户体验。

由于当前主流的浏览器都支持 HTML5 了，只有微软的 IE8 及之前的浏览器不支持 HTML5，因此依据趋势，我们在上传文件时，尽可能的采用 HTML5。在 HTML5 无法支持的情况下，我们才会采用传统的上传技术，但为了实现界面无刷新模式，需要将 form 的 target 指向一个 iframe，由这个 iframe 来接收 post 请求的返回页面。

由于 HYFileServer 是专用上传服务器，为了优化性能与正确返回信息，HYFileServer 需要提前知道每次上传的文件数量与上传的方式，上传数量是通过 form 的 filecount 字段给予，上传方式是通过 uploadway 字段给予，可以将这两个字段作为 hidden 类型附加在 form 里，如下所示：

```
<form id="form1" enctype="multipart/form-data" method="post" action="" target="iframe-upload-target">
   <input type="hidden" name="filecount" value="1" />
   <input type="hidden" name="uploadway" value="ajax" />
   <input type="file" name="fileToUpload" id="fileToUpload" onchange="fileSelected();"/>
</form>
```

如果上传时不带 filecount 字段，HYFileServer 服务器也能正常工作，但是会跟其它 web 服务器上传处理一样逐字节分析分析 multipart/form-data 数据来分割文件上传数据。uploadway 字段用于向服务器报告上传方式，可用的值为 iframe、ajax、flash 等等，只有用 iframe 上传方式时必须指定这个字段，否则服务器不能为你返回正确的信息。一般应该将 filecount 与 uploadway 放置在 form 的前段以便让 HYFileServer 能够提前检测到来启动优化处理过程。

3.2.1 HTML5 文件上传调用

HTML5 上传是采用 HTML5 的 FormData 与 XMLHttpRequest 来实现的，具体如何操作请查阅相关资料。

步骤如下：

1）创建 FormData 并将原始 form 表单的 DOM 元素作为一个参数传递给 FormData；

2）创建 XMLHttpRequest 对象并设置好参数

3）调用 XMLHttpRequest 对象的 open 方法并制定正确的 URL，这个 URL 应该是上传服务器的接受地址。

4）调用 XMLHttpRequest 的 send 方法。

示范代码如下

```
var myForm = document.getElementById("form1");
var fd = new FormData(myForm);

var xhr = new XMLHttpRequest();
xhr.upload.addEventListener("progress", uploadProgress, false);
xhr.overrideMimeType("application/octet-stream");
xhr.addEventListener("load", uploadComplete, false);
xhr.addEventListener("error", uploadFailed, false);
xhr.addEventListener("abort", uploadCanceled, false);
xhr.open("POST", upload_url,true);
xhr.send(fd);
```

其中 uploadComplete、uploadFailed、uploadCanceled 为自定义的回调函数，upload_url 为上传地址。HTML5 可以设置进度回调函数，在上传过程中可以显示进度，这对于大文件上传非常用用。我们还可以用这个回调函数来显示上传位率等信息。

所有移动智能设备都能采用 HTML5 方式上传文件。

注意：采用 HYFileServer 作为上传服务器，因为与应用系统不是在同一个域下面，浏览器需要跨域权限才能上传，HYFileServer 已经内部实现了跨域机制，调用者无需关心这方面问题。

3.2.2 IFRAME 文件上传调用

由于微软微软的 IE8 及以下浏览器不支持 HTML5，我们旧的采用传统的方式来上传文件。为了实现界面的优雅设计，即无刷新方式上传文件，在进行传统方式上传文件时，将 form 的 target 属性指向一个 iframe，浏览器将在上传完毕后将返回网页输出到这个 iframe，通过将这个 iframe 隐藏，可以避免破坏界面的美观。

iframe 方式上传文件的步骤如下：

1）构建一个隐藏的 iframe。

2）设置 form 的 target 为网页里的一个 iframe 的名字一边指向这个 iframe。

3）用普通的 submit 提交表单。

4）从 iframe 里获取返回信息

iframe 方式也适用于现代浏览器，包括移动设备浏览器。

采用 iframe 方式提交数据后，在数据提交结束并获得反馈后，会触发 iframe 的 load 事件，我们通过设置 ifame 的 load 事件来获得反馈。由于访问 iframe 与应用服务器仍旧是不同的域，需要绕过这个跨域访问限制，iframe 的 name 属性可以作为数据交互的纽带。HYFileServer 服务器通过产生相关代码来修改 iframe 的 name 属性，父框架通过访问 iframe 的 name 属性来获得反馈信息。

如果上传方式为 iframe，则 HYFileServer 服务器在上传成功后会返回类似如下信息：

```
<!DOCTYPE html><html><head><meta http-equiv="Content-type" content="text/html; charset=utf-8"/></head><body><div id="result-obj">
{
 "result":"success",
 "msg":"file upload success!",
 "files": [
  {
  "file": "Live Channel 120160415161429_1.mp4",
  "url": "/Uploads/videos/20170724/b7ec0cc079120b7cb62af53ff16db943.mp4",
  "path":"G:/Uploads/videos/20170724/b7ec0cc079120b7cb62af53ff16db943.mp4",
  "filesize":"8971797"
  }
 ]
}
</div></body>
<script language="javascript">
window.name=document.getElementById('result-obj').innerHTML;
</script></html>
```

具体返回的信息依据上传的文件而定。

返回的信息包括 javascript 调用，会修改 iframe 的 name，将 iframe 的 name 修改为文件上传信息，我们的父框架再访问 iframe 的 name 属性，由于这时的 iframe 的域已经与框架不同了，直接访问仍旧有跨域问题，我们通过将 iframe 的 src 指定为"about:blank"来切换 iframe 的域与框架为同一域来实现访问。

实现 iframe 跨域访问的伪代码如下：

```
var state = 1;
addEvent(fileUploadFrame,'load',function () {
var result;
//这里的代码是为了实现跨域访问 iframe 的 name 属性，IE8 目前可能没有跨域问题，但 Chrome、Firefox 等浏览器有
```

```
var iframe = fileUploadFrame;

if(state === 1) {
    result = iframe.contentWindow.name;
}
else if(state === 0){
    var except = 0;

    state = 1;
    var agent = window.navigator.userAgent.toLowerCase();

    if (agent.match(/msie/i)) {
        //如果是 IE，能够执行到这里，一般都是 IE8 或者以下，尝试获取 iframe 的 name
        try {
            result = iframe.contentWindow.name;
        }
        catch(e) {
            except = 1;
        }
    }

    if (except || typeof(result) == 'undefined') {
        try {
            iframe.contentWindow.location = "about:blank";
        }
        catch(e) {

        }
        return;
    }
}

//保存或显示最后结果
show_upload_result(result);
});
```

3.2.3 iframe 上传进度如何显示

通过 iframe 上传单靠浏览器是无法显示上传进度的，由于 HYFileServer 是专用上传服务器，在设计时已经考虑到这一点了。

解决办法如下：

1）在上传的 url 后附加 fileid 参数，fileid 的值随便指定，只要能够做到系统唯一即可，一般用当前计算机的时间戳值，由于这个值精确到毫秒，因此基本能够保证唯一。

举例说明：

例如假设上传文件的 url 为 http://192.168.1.201:8080/upload/video
采用 iframe 方式上传文件的 url 则变为
http://192.168.1.201:8080/upload/video?fileid=15783456788

2）通过上传服务器的/getprogress 从服务器上获得上传进度信息

获取进度信息的 URL 如下（假设上传地址为：http://192.168.1.201:8080/upload/video）：
http://192.168.1.201:8080/getprogress?fileid=15783456788

在上传服务器内部，由于 fileid 已经是唯一的了，因此可以直接用/getprogress 路径来获取进度，不需要担心分类目录问题。在多用户并发上传时即使理论上 fileid 的值可能冲突，但这种情况很少，可以忽略。

在启动上传后，通过 setInterval 函数设置的时钟事件来定期查询进度。

通过 /getprogress 获得的信息类似如下：

```
{"total": "128733821","finish": "31227904"}
```

其中 total 表示总的文件上传数据长度，finish 表示已经上传的文件数据长度。

3.2.4 其它上传方式

HYFileServer 服务器也支持 swfupload 等 flash 上传组件、jQuery uploader 等上传组件进行上传，如何调用第三方的浏览的上传组件请参考相关文档。

3.2.5 样本代码

实现文件上传的样本代码在 HYFileServer 目录的 html 子目录里，
upload.html ： 文件展示了如何自适应 html5 与 iframe 上传文件。
upload_file.html ： 非常简单的文件上传。
h5upload.html ： 展示了一个带有进度条的上传界面。

由于采用 HYFileServer 文件上传的技术与其它 HTTP 上传没有不同，因此，在熟悉 HTML 与 javascript 开发后界面可以任意创建。

将以上网页放置到任意 HTTP 的虚拟目录下即可工作。

注意修改上传地址，上传地址应该用实际的服务器地址。

upload.html 修改如下行

var upload_url = "http://192.168.0.139:8080/upload/video";

upload_file.html 与 h5upload.html 文件注意修改 action 的内容。

如何创建更美观的上传界面，请参考 HTML 的相关技术。

四、常见问题