# 工作平台多任务断点下载设计

一、简介

根据工作平台文件管理的讨论需求，参考360的云盘下载管理功能，下载要求需要单文件下载和文件夹整体下载，需要统计总下载进度和单个文件正在下载的进度，可以设置整体最大的下载速度，和并行下载的任务数。

所以此功能以多任务多线程的设计思路进行扩展，设计出以下满足需要的功能模块。

二、主要模块划分

1）任务管理模块(包括配置信息)

功能点：1、添加任务

2、取消任务

3、暂停任务

4、恢复任务

5、查询任务(包括任务数量、任务状态、任务进度、任务平均下载速度等)

6、获取任务总体进度

7、获取任务线程数量

8、设置同时任务最大数

9、设置默认线程数

10、获取任务ID

11、……

2) 单任务下载模块

功能点：1、添加下载任务

2、执行下载任务

3、任务是否在执行中

4、获取线程数(总数)

5、获取正在运行的线程数

6、获取任务状态

7、获取任务ID

8、获取文件长度

9、获取已完成字节数(续传使用)

10、获取文件名

11、…….

2) 多线程下载模块

功能点：1、多线程断点续传

2、URL解析重定向

3、获取文件总长度

4、获取文件当前下载进度

5、获取平均下载速度

6、添加任务到下载项

7、设置最大速度限制

8、MD5文件完整性校验

9、……

三、设计思路

1）首先，上层UI显示的信息关键是下载进度、下载速度、完成状态等一些需要让用户能够看到的内容。

2）其次，上层UI设置控制有限制下载的最大速度、最大同时下载的任务、调整下载任务的优先级、可以暂停、恢复、取消任务等。

3）最后，根据上层传来的FILEID当做唯一对应的任务参数，查询每个任务的具体信息等。

四、调用关系图

下载任务控制设置封装

多线程断点续传下载逻辑封装

IO操作封装、控制操作封装、线程操作封装等

五、部分定义以及组合关系

//定义大文件尺寸读取类型

typedef int int32;

typedef unsigned int uint32;

typedef \_\_int64 int64;

typedef unsigned \_\_int64 uint64;

//DWORD类型低位和高位取值使用

#define LOINT32(i) ((int)((\_\_int64)(i) & 0xffffffff))

#define HIINT32(i) ((int)((\_\_int64)(i) >> 32))

//单任务状态

enum \_SINGLE\_TASK\_STATUS

{

\_TASK\_NONE,//初始状态

\_TASK\_WAITTING,//等待中

\_TASK\_RUNNING,//下载中

\_TASK\_ERROR,//下载错误

\_TASK\_FINISH,//完成

};

//为以后扩展协议

enum \_PROTYPE

{

UNKNOW\_PROTOCOL = 0,//未知协议

HTTP\_PROTOCOL = 1,//HTTP协议

};

//业务层显示任务状态，以及任务文件类型分类

enum \_TASK\_TYPE{

\_STATUS\_ROOT,//初始化状态

\_STATUS\_RUNNING,//下载中

\_STATUS\_WAITING,//等待中

\_STATUS\_PAUSE,//暂停状态

\_STATUS\_WRONG,//下载错误

\_STATUS\_FINISH,//完成

\_STATUS\_FINISH\_SOFT,//软件类型分类，比如.exe

\_STATUS\_FINISH\_PACK,//压缩包类型分类,比如.zip

\_STATUS\_FINISH\_MUSIC,//音乐类型分类,比如.mp3

\_STATUS\_FINISH\_ISO,//镜像光盘类型分类,比如.iso

\_STATUS\_FINISH\_MOVIE,//电影类型分类，比如.rmvb

\_STATUS\_FINISH\_PUBLISH,//出版类型分类,比如.pdf

\_STATUS\_FINISH\_PICTURE,//图片类型分类,比如.bmp

\_STATUS\_FINISH\_UNKNOW,//未知下载文件类型,根据自定义类型以外的类型属于未知类型。

\_STATUS\_DELETE,//删除任务

};

//任务数据结构定义

struct \_TaskAttr

{

int nID;//任务ID，查询任务使用

\_TASK\_TYPE nStatus; //显示任务状态类型

int nThread;//线程数

int nPriority; //优先级

bool bAutoMd5;//MD5自动检测

std::string sName;//文件名称,不带路径

int64 nLen; //文件大小

float fRatio; //下载速率k/s

int64 nFinish; //完成大小

int nSpeed;//下载速度

int nTime;//完成时间

int nError; //错误时间

int nRetryTime; //重试等待时间

int nRetryWait; //重试次数

std::string sURL; //主URL

std::string sRefer; //重定向URL

std::vector<std::string> sMirrorURL; //镜像下载URL，提升下载速度。当前云盘业务用不到

std::string sSavePath;//保存路径

std::string sHttpProxy;//HTTP代理

std::string sMD5;//文件MD5

};

//显示图标状态

enum \_MSGTYPE {

MSG\_ERROR,//错误

MSG\_WARNNING,//警告

MSG\_INFO,//正常

MSG\_SUCCESS,//成功

MSG\_OUT,//上传

MSG\_IN,//下载

};

//系统日志信息

struct \_syslog

{

int ntime;//当前日志时间

std::string info;//日志信息

};

//任务日志信息

struct \_tasklog

{

int ntaskid;//任务id

int ntime;//当前日志时间

\_MSGTYPE ntype;//日志消息状态

std::string info;//日志内容

};

//代理设置

struct \_ProxyAttr

{

int nType; //HTTP,HTTPS,FTP....

std::string sName; //自定义名称，必须唯一

std::string sServer; //代理服务器名称

int nPort;//端口

std::string sUser;//用户名

std::string sPass;//密码

};

//单任务下载类设计，暂且定义这些接口

class CSingleTask

{

public:

CSingleTask(

CTaskManager \*parent,

std::string url,

std::string savepath,

std::string savename,

int taskid,

int ants = 5,

int retry = 10,

int retrywait = 60,

std::string refer = ""

);

virtual ~ CSingleTask ();

void AddMirrorUrl( std::string url );

bool Go();

bool IsRunning();

int GetThreadNum();

int GetRunningThreadNum();

\_SINGLE\_TASK\_STATUS GetStatus();

int GetTaskID();

int64 GetFileLen();

void GetFinishByte(int64& thistime, int64& lasttime );

std::string GetFilename();

std::string GetTaskUrl();

CTaskManager\* GetParent();

void ThreadQuit( int threadnum );

void GetRunningArgs( float& fratio, int64& finish, int& speed, int& ttf );

void LogError();

protected:

static void\* SwapInitThread( void\* );

void FinishTask( \_SINGLE\_TASK\_STATUS status = \_TASK\_RUNNING );

void InitThread();

int m\_nTaskID;

int m\_nStage;

private:

\_SINGLE\_TASK\_STATUS m\_nStatus;

CTaskManager \*m\_pParent;

int m\_nAnts;

int m\_nRunningAnts;

int m\_nRetry, m\_nRetryWait;

int64m\_nLastFinishByte;

std::string m\_sSavePath;

std::string m\_sUrl;

std::string m\_sFilename;

int64m\_nFileLen;

std::vector<std::string> m\_sMirrorUrl;

struct \_task\_info

{

int time;

\_MSGTYPE type;

std::string info;

};

};

//任务管理类设计

class CTaskManager

{

public:

CTaskManager();//构造参数如果需要可以定义一个对象指针

~CTaskManager();

std::string GetTaskFilename( int ntask );

bool AddTask( \_TaskAttr& task );

bool RemoveTask( int ntask );

void AddTaskMirrorUrl( int ntask, std::string mirror );

int RemoveAllTask();

float GetTaskProgress( int ntask ); //获得任务总体进度

int64 GetTaskSize( int ntask ); //文件大小

std::string GetTaskUrl( int ntask );

int64 GetFinishByte( int ntask, int& speed ); //文件完成大小

\_SINGLE\_TASK\_STATUS GetTaskStatus( int ntask ); //任务当前进行状态，完成，下载中，停止中

int GetTaskNumber();

int GetTaskThreadNum( int ntask );

int GetErrorTime( int ntask );

int SetDefaultTaskNum(int num); //设置最大运行任务数量，大于这个则排队

int SetDefaultThreadNum(int num); //设置缺省线程数量，不影响当前的下载任务

int SetTaskPriority(int ntask,int priority); //设置和调整任务优先级

//完成比例，完成数量，平均速度，完成时间，4个动态数据一个函数提取，用在任务显示更新上

bool GetRunningArgs( int ntask, float& fratio, int64 & finish, int& speed, int& ttf );

bool ReadTaskDB();

bool SaveTaskDB();

bool SaveProxyDB();

bool ReadProxyDB();

bool ReadConfig();

bool SaveConfig();

protected:

bool DeleteTask( int ntask );

int StopTask(int ntask);

int CancelTask(int ntask);

int PauseTask(int ntask);

int ResumeTask(int ntask);

private:

CSingleTask \* GetTaskObj( int ntask );

int m\_nDefaultMaxTask; //最大同时任务

int m\_nDefaultThreadNum;

std::vector< CSingleTask \*> m\_TaskList; //任务表

std::string m\_sSavePath;

int m\_nTaskUI; //需要更新显示的任务ID，-1就是没有需要更新的

};

五、调研成果

1）技术选型

**CURL开源库介绍：**

curl是利用URL语法在命令行方式下工作的开源[文件传输](http://baike.baidu.com/view/543341.htm)工具。它被广泛应用在Unix、多种Linux发行版中，并且有DOS和Win32、Win64下的移植版本。[

它支持很多协议：FTP, FTPS, HTTP, HTTPS, GOPHER, TELNET, DICT, FILE 以及 LDAP。

curl同样支持HTTPS认证，HTTP POST方法， HTTP PUT方法， FTP上传， kerberos认证，HTTP上传，[代理服务器](http://baike.baidu.com/view/751.htm)， cookies， 用户名/密码认证， 下载文件[断点续传](http://baike.baidu.com/view/41621.htm)，上载文件断点续传，http代理服务器管道（ proxy tunneling）， 甚至它还支持IPv6， socks5代理服务器，通过http代理服务器上传文件到[FTP服务器](http://baike.baidu.com/view/106754.htm)等等，功能十分强大。

Windows[操作系统](http://baike.baidu.com/view/880.htm)下的[网络蚂蚁](http://baike.baidu.com/view/4051.htm)，[网际快车](http://baike.baidu.com/view/4090.htm)(FlashGet)的功能它都可以做到。准确的说，curl支持文件的上传和下载，所以是一个综合传输工具，但是按照传统，用户习惯称curl为[下载工具](http://baike.baidu.com/view/336014.htm)。

curl是[瑞典](http://baike.baidu.com/view/14445.htm)curl组织开发的，您可以访问: http://curl.haxx. se/ 获取它的[源代码](http://baike.baidu.com/view/60376.htm)和相关说明。

此库需要自己从每个函数接口自己定制，代码容易懂，控制容易，但需要自己定制多任务和下载多线程模块开发。

有人维护，文档很全（比较偏向这个技术选型，以后添加新协议支持容易扩展）

**Multiget开源库介绍：**

MultiGet 是一个图形化界面的下载管理器，其主要特性包括：  
  
\* 支持 http、ftp 协议；  
\* 支持多线程、多任务下载文件；  
\* 支持断点续传；  
\* 支持 SOCKS、ftp、http 代理；  
\* 支持悬浮窗拖拉操作；  
\* 支持剪贴板自动监视；  
\* 支持系统托盘；  
\* 支持英文、中文界面；

此开源库大部分功能都支持，但在其上面开发自己的业务代码需要重新整理，代码有点凌乱不易控制。

文档资料不全。2008年后不更新了。

**金山开源软件管理代码：**

代码较多，支持功能也多，第三方库引入较多，但缺少重定向等功能，需要自己根据代码弥补。代码较复杂，不容易短时间上手。没有开发文档资料！

代码已经不更新。