Factory Reset时备份目录

#若要备份data分区的数据，需关闭data分区的加密：

修改 device/mediatekprojects/xxxx/ProjectConfig.mk

MTK\_ENCRYPTION\_DEFAULT\_OFF = yes

MTK\_ENCRYPTION\_FDE\_TO\_FBE = no

MTK\_ENCRYPTION\_TYPE\_FILE = no

#修改 /bootable/recovery/recovery.cpp

1、添加头文件

+#include<dirent.h>

+#include<sys/stat.h>

2、添加function

@@ -187,6 +190,124 @@

\* 7b. the user reboots (pulling the battery, etc) into the main system

\*/

+#if 1

+

+int is\_dir(char\* path){ //判断是否为目录

+ struct stat st;

+ stat(path,&st);

+ if(S\_ISDIR(st.st\_mode)){

+ return 1;

+ }

+ else{

+ return 0;

+ }

+}

+

+int endwith(const char\* s,const char c){ // 判断结尾是否为C

+ if(s[strlen(s)-1]==c){

+ return 1;

+ }

+ else{

+ return 0;

+ }

+}

+char\* str\_contact(const char\* str1,const char\* str2){ //对字符串进行连接

+ char\* result = NULL;

+

+ result=(char\*)malloc(strlen(str1)+strlen(str2)+1);

+

+ memset(result, 0 ,sizeof(result));

+ strcat(result,str1);

+ strcat(result,str2);

+ return result;

+}

+char\* str\_contact(char\* str1,const char\* str2){ //对字符串进行连接

+ strcpy(str1,str2);

+ return str1;

+}

+

+bool copy\_file(const char\* source\_path,const char \*destination\_path){ //复制文件

+ char buffer[1024];

+ FILE \*in,\*out;

+

+ printf("mxd copy\_file\n");

+ if((in=fopen(source\_path,"r"))==NULL){

+ printf("failed to open %s！\n", source\_path);

+ return false;

+ }

+ if((out=fopen(destination\_path,"w+"))==NULL){

+ printf("failed to open %s！\n", destination\_path);

+ return false;

+ }

+ int len;

+ while((len=fread(buffer,1,1024,in))>0){

+ fwrite(buffer,1,len,out);

+ }

+ chmod(destination\_path,0666);

+ fclose(out);

+ fclose(in);

+ return true;

+}

+void copy\_folder(const char\* source\_path,const char \*destination\_path){ //复制目录

+ DIR\* dp\_d = opendir(destination\_path);

+ if(!dp\_d){

+ if (mkdir(destination\_path,0777))

+ {

+ printf("failed to open %s！\n", destination\_path);

+ }

+ }

+ closedir(dp\_d);

+ printf("mxd source\_patch = %s, destination\_path = %s\n", source\_path, destination\_path);

+#if 0

+ char \*path;

+ path = (char\*)malloc(512);

+ memset(path,0,sizeof(path));

+ path = str\_contact(path,source\_path);

+#endif

+ struct dirent\* filename;

+ DIR\* dp=opendir(source\_path);

+

+ while(1){

+ filename=readdir(dp);

+ if (filename) {

+

+ printf("mxd filename->d\_name = %s\n", filename->d\_name);

+ char \*file\_source\_path;

+ //file\_source\_path=(char\*)malloc(512);

+ //memset(file\_source\_path,0,sizeof(file\_source\_path));

+

+ char \*file\_destination\_path;

+ //file\_destination\_path=(char\*)malloc(512);

+ //memset(file\_destination\_path,0,sizeof(file\_destination\_path));

+

+ file\_source\_path=str\_contact(source\_path,"/");

+ file\_source\_path=str\_contact(file\_source\_path,filename->d\_name);

+

+ file\_destination\_path=str\_contact(destination\_path,"/");

+ file\_destination\_path=str\_contact(file\_destination\_path,filename->d\_name);

+

+ printf("mxd file\_source\_path = %s, file\_destination\_path = %s\n", file\_source\_path, file\_destination\_path);

+ if(is\_dir(file\_source\_path)){

+ if(!endwith(file\_source\_path,'.')){

+ copy\_folder(file\_source\_path,file\_destination\_path);

+ }

+ } else {

+ copy\_file(file\_source\_path,file\_destination\_path);

+ printf("copy %s to %s success！\n",file\_source\_path,file\_destination\_path);

+ }

+ free(file\_source\_path);

+ free(file\_destination\_path);

+ } else {

+ break;

+ }

+ }

+ closedir(dp);

+}

+#endif

+

+void store\_data(const char\* source\_path, const char\* destination\_path) //参数1为源目录，参数2为目标目录

+{

+ ensure\_path\_mounted("/data");

+ copy\_folder(source\_path, destination\_path);

+ ensure\_path\_unmounted("/data");

+}

3、在wipe\_data函数中调用备份function

@@ -754,7 +875,9 @@

static bool wipe\_data(Device\* device) {

modified\_flash = true;

+ store\_data("/data/app", "/tmp/app");

ui->Print("\n-- Wiping data...\n");

bool success =

device->PreWipeData() &&

erase\_volume("/data") &&

@@ -762,6 +885,7 @@

(has\_nvdata ? erase\_volume(NVDATA\_ROOT) : true) &&

device->PostWipeData();

ui->Print("Data wipe %s.\n", success ? "complete" : "failed");

+ store\_data("/tmp/app", "/data/app");

return success;

}