Windows Internals II
Project II: Writing OS subsystems 補足資料

金田憲二
kaneda@is.s.u-tokyo.ac.jp

平成 17 年 1 月 27 日

概 要

この課題では，NT オブジェクトマネージャの提供するオブジェクトを扱う．具体的には，ユーザモードプロセスとして走る OS サービスを記述する．Native NT API を簡略化した OZ ライブラリが提供されているので，それを利用する．

1 開発，実行環境の設定

Project I の補足資料を参考にすること．

2 デモサブシステムの実行

2.1 概要

単純なサービスを提供するサーバを実行する．このサーバは，クライアントからプリントリクエストを受付け，送られてきた文字列を端末に表示する．ソースコードは，

`http://www.i.u-tokyo.ac.jp/ss/msprojects`

から取得できる．

`Porject2\OZServer` 以下にある `ozserver.c` が，サービスを提供するサーバ (OZServer) のソースコードとなっている．

`Porject2\OZApp` 以下にある `ozapp.c` が，そのサービスを呼び出すクライアント (OZApp) のソースコードとなっている．

2.2 ビルド・実行

この OZServer と OZApp のビルド・実行は，以下のようにして行う．

1. Project I の補足資料の 1.2 節にある手順に従ってコマンドプロンプトを立ち上げる.

2. Project2 ディレクトリに移動し,

```
build
```

と入力し，ビルドを行う.

3. ビルドされた ozserver.exe，ozapp.exe と，LIB 以下にある OZ ライブラリを同一ディレクトリ（例えば BIN ディレクトリの下）に移動させる．なお，以上の 2. と 3. の手順を一括して行いたい場合には，以下のようなスクリプトを用意しておけばいい.

```
build

copy /Y OZApp\objchk_wxp_x86\i386\ozapp.exe bin\
copy /Y OZApp\objchk_wxp_x86\i386\ozapp.pdb bin\

copy /Y OZServer\objchk_wxp_x86\i386\ozserver.exe bin\
copy /Y OZServer\objchk_wxp_x86\i386\ozserver.pdb bin\

copy /Y LIB\OZUTokyo2004.dll bin\
copy /Y LIB\OZUTokyo2004.lib bin\
copy /Y LIB\OZUTokyo2004.pdb bin\
```

4. 実行ファイルの移動先のディレクトリに移動して

```
ozserver.exe
```

と入力し，アプリケーションを実行する.

```
>ozserver.exe
Running OZApp.exe
************************
got this message?
************************
bye
```

と端末に出力されれば成功.

2.3 ソースコードの概要

OZServer（ozserver.c）は，まず，プロセスの生成を行う（CreateProcess 関数）．このプロセスが ozapp.exe を実行する．そのあと，LPC ポートでサービスを受付け，処理する（OzDispatchSyscalls 関数）．

OZApp（ozapp.c）は，OzSyscall 関数を呼び出し，myos_printmsg（= 0）というコード値のサービスを OZServer に要求する．

OZServer は，この要求を受けると，mysyscall_printmsg 関数を実行する．この関数は，サービスの第一引数で指定されたアドレスから，第二引数で指定されたバイト長だけデータを読み込み．その読み込んだデータを端末に印字する．この際に，OZApp 側のメモリから読み込みを行うために，OzReadChildMemory を用いる．

2.4 ソースコード中で使用される Windows API

2.4.1 CreateProcess

新しくプロセスを生成する．新しく生成されたプロセスは，指定された実行可能ファイルを実行する．

```
BOOL CreateProcess(
  LPCTSTR lpApplicationName, // 実行可能モジュールの名前
  LPTSTR lpCommandLine, // コマンドラインの文字列
  LPSECURITY_ATTRIBUTES lpProcessAttributes,
                                  // セキュリティ記述子
  LPSECURITY_ATTRIBUTES lpThreadAttributes,
                                  // セキュリティ記述子
  BOOL bInheritHandles, // ハンドルの継承オプション
  DWORD dwCreationFlags, // 作成のフラグ
  LPVOID lpEnvironment, // 新しい環境ブロック
  LPCTSTR lpCurrentDirectory,  // カレントディレクトリの名前
  LPSTARTUPINFO lpStartupInfo, // スタートアップ情報
  LPPROCESS_INFORMATION lpProcessInformation // プロセス情報
);
```

パラメータ

lpApplicationName 実行するモジュールの名前を指定する．

lpCommandLine 実行するコマンドラインを指定する．NULL を指定した場合，lpApplicationName パラメータで指定した文字列をコマン

ドラインとして扱う.

lpProcessAttributes SECURITY_ATTRIBUTES 構造体へのポインタを渡し，子プロセスがハンドルを継承できるかどうかを指定する．NULL を指定した場合は，継承不可能となる.

lpThreadAttributes SECURITY_ATTRIBUTES 構造体へのポインタを渡し，子プロセスがハンドルを継承できるかどうかを指定する．NULL を指定した場合は，継承不可能となる.

bInheritHandles 新しいプロセスが，呼び出し側プロセスのハンドルを継承するかどうかを指定する.

dwCreationFlags 優先順位クラスとプロセスの作成を制御するフラグを指定する．ResumeThread 関数を呼び出すまでスレッドを開始させない場合，CREATE_SUSPENDED を指定.

lpEnvironment 新しいプロセスの環境ブロックへのポインタを指定する．呼び出し側プロセスと同じ環境を使う場合，NULL を指定.

lpCurrentDirectory 新しいプロセスのカレントドライブ名とカレントディレクトリ名を指定する.

lpStartupInfo 新しいプロセスのメインウィンドウの表示方法を指定する.

lpProcessInformation PROCESS_INFORMATION 構造体へのポインタを指定する．関数から制御が返ると，この構造体に，新しいプロセスに関する情報が格納される.

戻り値 関数が成功すると 0 以外の値が返り，失敗すると 0 が返る.

2.4.2 RtlZeroMemory

指定されたメモリブロックを 0 で埋める.

```
VOID
  RtlZeroMemory(
    IN VOID UNALIGNED  *Destination,
    IN SIZE_T  Length
    );
```

パラメータ

Destination 0 で埋めるメモリのアドレスを指定する.

Length バイト長を指定する.

2.5 ソースコード中で使用する OZ ライブラリの API

API の一部を簡単に説明する. より詳しい情報を知りたい場合は, ヘッダファイル INC\OZ-UTokyo2004.h などを参照すること.

2.5.1 OzSyscall

(クライアントが) サービスを呼び出す.

```
OZDLL_IMPORT
ULONG_PTR
WINAPI
OzSyscall (
    enum mysyscode ozsyscode,
    ULONG_PTR      arg0,
    ULONG_PTR      arg1,
    ULONG_PTR      arg2,
    ULONG_PTR      arg3
    );
```

パラメータ

ozsyscode サービスのコード値を指定する. OZ-UTokyo2004.h 中の enum mysyscode で定義されている値.

arg0, arg1, arg2, arg3 サービスに渡す引数.

戻り値

2.5.2 OzReadChildMemory

(サーバが) クライアントのメモリ領域からデータを読み込む.

```
OZDLL_IMPORT
BOOL
WINAPI
OzReadChildMemory (
    IN HANDLE  childprocess,
    IN PVOID   childaddress,
    IN  SIZE_T buffersize,
    OUT PVOID  buffer
    );
```

パラメータ

childprocess クライアントプロセスへのハンドル.

childaddress 読み込み先の (クライアントの) アドレス.

buffersize 読み込むデータのバイト数.

buffer 読み込んだデータを格納する (サーバの) バッファへのポインタ.

3 IO サービスの追加

3.1 概要

OZServer を拡張する. 具体的には, 以下のような UNIX の IO システムコールとほぼ同様のサービスを提供するようにする.

```
int fd = open(char *path);
int read(fd, buffer, len);
void close(fd);
```

3.2 新しいサービスの追加方法

ソースに以下のような変更を加えることによって, 新たなサービスを追加することができる.

3.2.1 OZ-UTokyo2004.h に加える変更

まず,

```
enum mysyscode
{
      myos_printmsg  = 0
//   ,myos_syscall1  = 1
//   ,myos_syscall2  = 2
//   ,myos_syscall3  = 3
//   ,myos_syscall4  = 4

};
```

という箇所に，自分が新たに追加したいサービスのコード値（ユーザがサービス呼び出し時に指定する整数値）を追加する．

次に，

```
#define MYSYS_LIMIT 1
```

で定義されているサービス数を変更する．

3.2.2 ozserver.c に加える変更

サービスを処理する関数を設定する．

```
POZSYSCALLHANDLER syscallhandlers[MYSYS_LIMIT] = {
    mysyscall_printmsg    // myos_printmsg
};
```

という箇所に，サービスを処理する関数名を追加する．クライアントがコード値 c のサービスを呼び出した場合，サーバ側では，syscallhandlers[c] に格納された関数が呼び出されるようになっている．

サービスを処理する関数の型は，以下のようになっている．

```
ULONG_PTR mysyscall(HANDLE clientprocess,
                    POZSYSCALL syscall)
```

第一引数は，クライアントプロセスへのハンドル．第二引数は，クライアントがサービス呼び出し時に指定した引数などの情報．POZSYSCALL は以下のような型となっており，例えば配列 ozarg には引数の情報が格納される．

```
typedef struct _OZSYSCALL
{
    enum mysyscode  ozsyscode;
    ULONG_PTR       ozarg[4];
    ULONG_PTR       result;
} OZSYSCALL, *POZSYSCALL;
```

また,

```
PCHAR mysyscallname[MYSYS_LIMIT] = {
    "printmsg"
};
```

に新たなサービスの名前を追加しておく.

注意　サービスの引数としてポインタが渡される場合，そのポインタのさすアドレスは，サーバではなくクライアントのメモリのアドレスであり，直接サーバから読むことはできない.

その際には，OzReadChildMemory 関数や OzWriteChildMemory 関数を用いて，サーバはクライアントのメモリを読み書きする必要がある.

3.3　ソースコード中で使用する OZ ライブラリの API

3.3.1　OzWriteChildMemory

サーバが，クライアントのメモリ領域にデータを書き込む.

```
OZDLL_IMPORT
BOOL
WINAPI
OzWriteChildMemory (
    IN HANDLE  childprocess,
    IN PVOID   childaddress,
    IN  SIZE_T buffersize,
    OUT PVOID  buffer
    );
```

パラメータ

childprocess クライアントプロセスへのハンドル.

childaddress 書き込み先の (クライアントの) アドレス.

buffersize 書き込むデータのバイト数.

buffer　データを格納する (サーバの) バッファへのポインタ.

この OzWriteChildMemory 関数は，open の第一引数のファイル名を取得する際や，ファイルから読み込んだ内容をクライアントが指定したバッファに書き込む際に利用される.

目 次