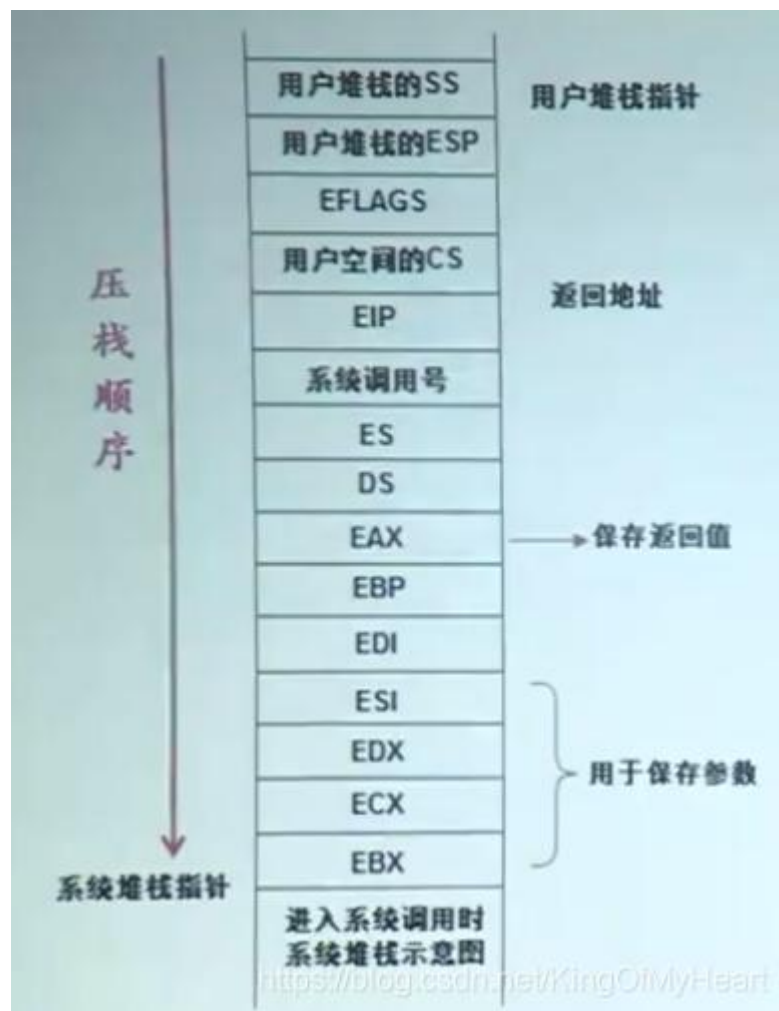


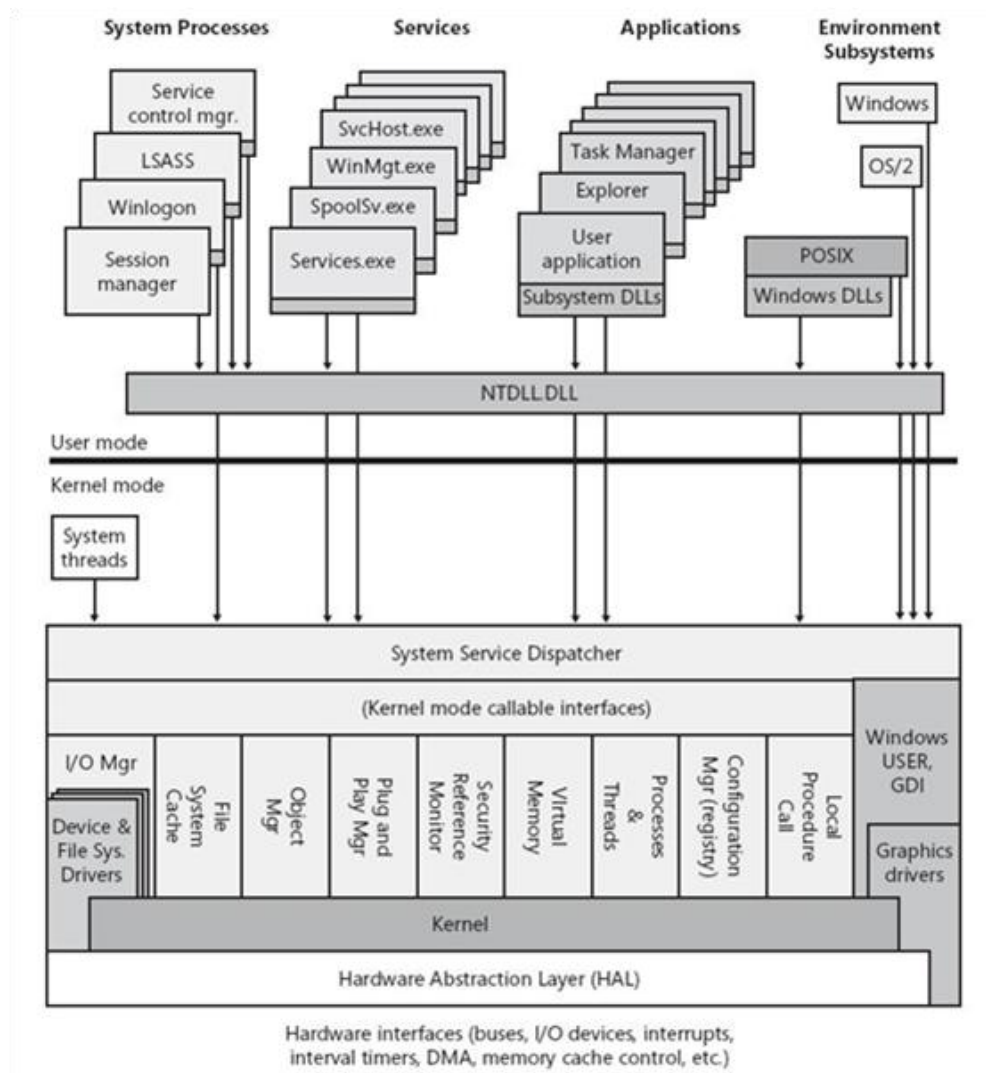
脱机命令接口（批处理脚本）

```
@echo off
set rarpath="c:\program files\winrar\"
set logpath=c:\log\
set prefix=anc
rem 获取昨天日期
set YE=%date:~0,4%
set MO=%date:~5,2%
.....
set Han=%LY%-%LM%-%LD%
echo 昨天的日期为: %Han%
%logpath:~0,2%
cd %logpath%
%rarpath%rar.exe a %prefix%_server_%Han%.rar server_%Han%.log
```

系统调用时的内核栈



Windows架构



A Simplest Win32 Program

- Important basic concepts
 - Message & message queue
 - A structure sent by OS, to inform the application about some events
 - Handle of window, indicating which window the message is sent to
 - Message ID, indicating what kind of message this is
 - wParam & lParam, two 32-bit parameters, different use for different message
 - Each UI thread has a message queue (maintained by OS), its program can get a message through the API function GetMessage()
 - A Windows application does not execute and quit immediately, the main task throughout its life is to process messages
 - In order to respond to the messages generated from time to time, an application must include a “message loop” in its program

A Simplest Win32 Program

- Important basic concepts
 - Callback function
 - A function written by the application programmer, whose pointer (address) is passed to OS by the programmer, and called by OS
 - Handle
 - Basically, an integer, an ID number identifying some system object
 - System object: something belonging to the OS, but may be operated by the applications, for example, a thread, a window, a menu, a button, a screen area, ...
 - For security consideration, the OS cannot pass the pointer (address) of a system object to the applications, instead, it passes an ID
 - Taking a handle as parameter, an application can operate on a system object by calling APIs

A Simplest Win32 Program

- Model

Simply a window
frame, no content

Hi, Windows, I have a window like this ...

The messages for my window should be processed by this guy (WndProc)

Hi, Windows, I have finished the pre-processing of the message

WinMain()

- Register Window
- Create Window
- Show Window

Message Loop

↓ WndProc(hwnd, ...)

```
switch(message) {
```

Callback PAINT:
ent area
Paint something

GetMessage

Application

Windows

msg queue

A Simplest Win32 Program

```
#include <windows.h>
LONG WINAPI WndProc (HWND, UINT, WPARAM, LPARAM);

int WINAPI WinMain (HINSTANCE hInstance, HINSTANCE hPrevInstance,
    LPSTR lpszCmdLine, int nCmdShow)
{
    WNDCLASS wc;
    HWND hwnd;
    MSG msg;

    wc.style = 0;
    wc.lpfnWndProc = (WNDPROC) WndProc;
    wc.cbClsExtra = 0;
    wc.cbWndExtra = 0;
    wc.hInstance = hInstance;
    wc.hIcon = LoadIcon (NULL, IDI_WINLOGO);
    wc.hCursor = LoadCursor (NULL, IDC_ARROW);
    wc.hbrBackground = (HBRUSH) (COLOR_WINDOW + 1);
    wc.lpszMenuName = NULL;
    wc.lpszClassName = "MyWndClass";

    RegisterClass (&wc);
    .....
```

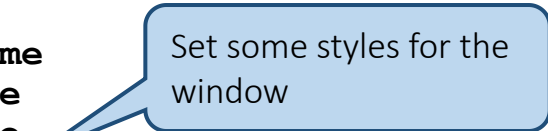
Set some styles for the window class

// Class style
// Window procedure
// Class extra bytes
// Window extra bytes
// Instance handle
// Icon handle
// Cursor handle
// Background color
// Menu name
// WNDCLASS name

A Simplest Win32 Program

```
.....
hwnd = CreateWindow (
    "MyWndClass",           // WNDCLASS name
    "SDK Application",      // Window title
    WS_OVERLAPPEDWINDOW,   // Window style
    CW_USEDEFAULT,          // Horizontal position
    CW_USEDEFAULT,          // Vertical position
    CW_USEDEFAULT,          // Initial width
    CW_USEDEFAULT,          // Initial height
    HWND_DESKTOP,           // Handle of parent window
    NULL,                   // Menu handle
    hInstance,              // Application's instance handle
    NULL,                   // Window-creation data
);
ShowWindow (hwnd, nCmdShow);
UpdateWindow (hwnd);

while (GetMessage (&msg, NULL, 0, 0)) {
    TranslateMessage (&msg);
    DispatchMessage (&msg);
}
return msg.wParam;
}
```



Set some styles for the window

A Simplest Win32 Program

```
LRESULT CALLBACK WndProc (HWND hwnd, UINT message, WPARAM wParam,
    LPARAM lParam)
{
    PAINTSTRUCT ps;
    HDC hdc;

    switch (message) {

    case WM_PAINT:
        hdc = BeginPaint (hwnd, &ps);
        Ellipse (hdc, 0, 0, 200, 100);
        EndPaint (hwnd, &ps);
        return 0;

    case WM_DESTROY:
        PostQuitMessage (0);
        return 0;
    }
    return DefWindowProc (hwnd, message, wParam, lParam);
}
```