Logging the foreground process as it changes

devblogs.microsoft.com/oldnewthing/20131202-00

December 2, 2013

Raymond Chen

Today's Little Program simply logs all changes to the foreground window by recording the path to the application the user switched to. You might use this as part of a usability study to monitor what applications users spend most of their time in.

Most of this code is just taking things we already know and snapping them together.

1. Using accessibility to monitor events, specifically to monitor foreground changes.
2. `GetWindowThreadProcessId` to get the process ID from a window.
3. `OpenProcess` to get a handle to a process given the process ID.
4. `QueryFullProcessImageName` to get the path to the application from the handle. (For Windows XP, you can use `GetProcessImageFileName`.)

Take our scratch program and make these changes:

```
BOOL QueryWindowFullProcessImageName(
    HWND hwnd,
    DWORD dwFlags,
    PTSTR lpExeName,
    DWORD dwSize)
{
 DWORD pid = 0;
 BOOL fRc = FALSE;
 if (GetWindowThreadProcessId(hwnd, &pid)) {
  HANDLE hProcess = OpenProcess(
          PROCESS_QUERY_LIMITED_INFORMATION, FALSE, pid);
  if (hProcess) {
   fRc = QueryFullProcessImageName(
          hProcess, dwFlags, lpExeName, &dwSize);
   CloseHandle(hProcess);
  }
 }
 return fRc;
}
```

The `QueryWindowFullProcessImageName` function is the meat of the program, performing steps 2 through 4 above.

Now we just hook this up in our event callback function. This should look really familiar, since we did pretty much the same thing earlier this year.

```
BOOL
OnCreate(HWND hwnd, LPCREATESTRUCT lpcs)
{
 g_hwndChild = CreateWindow(TEXT("listbox"), NULL,
     LBS_HASSTRINGS | WS_CHILD | WS_VISIBLE | WS_VSCROLL,
     0, 0, 0, 0, hwnd, NULL, g_hinst, 0);
 if (!g_hwndChild) return FALSE;
 return TRUE;
}
void CALLBACK WinEventProc(
    HWINEVENTHOOK hWinEventHook,
    DWORD event,
    HWND hwnd,
    LONG idObject,
    LONG idChild,
    DWORD dwEventThread,
    DWORD dwmsEventTime
)
{
 if (event == EVENT_SYSTEM_FOREGROUND &amp;
     idObject == OBJID_WINDOW &&
     idChild == CHILDID_SELF)
 {
  PCTSTR pszMsg;
  TCHAR szBuf[MAX_PATH];
  if (hwnd) {
   DWORD cch = ARRAYSIZE(szBuf);
   if (QueryWindowFullProcessImageName(hwnd, 0,
                     szBuf, ARRAYSIZE(szBuf))) {
    pszMsg = szBuf;
   } else {
    pszMsg = TEXT("<unknown>");
   }
  } else {
   pszMsg = TEXT("<none>");
  }
  ListBox_AddString(g_hwndChild, pszMsg);
 }
}
int WINAPI WinMain(HINSTANCE hinst, HINSTANCE hinstPrev,
                   LPSTR lpCmdLine, int nShowCmd)
{
 ...
  ShowWindow(hwnd, nShowCmd);
 HWINEVENTHOOK hWinEventHook = SetWinEventHook(
     EVENT_SYSTEM_FOREGROUND,
     EVENT_SYSTEM_FOREGROUND,
     NULL, WinEventProc, 0, 0,
     WINEVENT_OUTOFCONTEXT);
  while (GetMessage(&msg, NULL, 0, 0)) {
   TranslateMessage(&msg);
   DispatchMessage(&msg);
```

```
  }
  if (hWinEventHook) UnhookWinEvent(hWinEventHook);
...
}
```

The main program installs an accessibility hook for the `EVENT_SYSTEM_FOREGROUND` event, and each time the event fires, it extracts the process name and logs it to the screen. Since the notification is asynchronous, the foreground window may have been destroyed by the time the notification is received, so we have to be prepared for that case.

Raymond Chen

Follow