

This is to test whether some 'outside' code is not rewriting the cache:

```
HFONT GetWin32Font(Font fnt, int angle)
{
    LTIMING("GetWin32Font");
    static HFontEntry cache1[1000000];
    static HFontEntry cache[FontCACHE];
    static HFontEntry cache2[1000000];
    ONCELOCK {
        for(int i = 0; i < FontCACHE; i++)
            cache[i].font.Height(-30000);
    }
    HFontEntry be;
    be = cache[0];
    for(int i = 0; i < FontCACHE; i++) {
        HFontEntry e = cache[i];
        if(i)
            cache[i] = be;
        if(e.font == fnt && e.angle == angle) {
            if(i)
                cache[0] = e;
            return e.hfont;
        }
        be = e;
    }
    LTIMING("GetWin32Font2");
    if(be.hfont)
        DeleteObject(be.hfont);

    be.font = fnt;
    be.angle = angle;
    be.hfont = CreateFont(
        fnt.GetHeight() ? -abs(fnt.GetHeight()) : -12,
        fnt.GetWidth(), angle, angle, fnt.IsBold() ? FW_BOLD : FW_NORMAL,
        fnt.IsItalic(), fnt.IsUnderline(), fnt.IsStrikeout(),
        fnt.GetFace() == Font::SYMBOL ? SYMBOL_CHARSET : DEFAULT_CHARSET,
        fnt.IsTrueTypeOnly() ? OUT_TT_ONLY_PRECIS : OUT_DEFAULT_PRECIS,
        CLIP_DEFAULT_PRECIS,
        fnt.IsNonAntiAliased() ? NONANTIALIASED_QUALITY : DEFAULT_QUALITY,
        DEFAULT_PITCH|FF_DONTCARE,
        fnt.GetFaceName()
    );
}
```

```
cache[0] = be;  
return be.hfont;  
}
```
