Robin Mills

DOCUMENT REVISION

Text:	January 2010
Photograph:	September 2009

Profile

I am a software engineer. I've been involved with computer software since school. I've worked on many different technologies including:

Development of Compilers and Interpreters	Windows, Mac, Linux/Unix, Embedded & Mobile.
Computer Graphics	Computer Aided Design
Build Engineering. Documentation. Scripting.	Rapid Prototyping and Demonstration Products.

My strongest asset is explaining things and I've lost count of the folks who've said: "When you explain things, it always sounds so simple." My reply: "Thank You. It works because it is simple!".

Experience

CONTRACT ENGINEER, NOVARIANT INCORPORATED, FREMONT, CALIFORNIA, SEP 08 - DEC 09

All purpose engineer for the company's embedded steering controller. This is a robot to drive a farm tractor. We've shipped this on both Windows/CE and Linux/Embedded. I initially ported the core code (1 million lines of Linux C++) to Windows. I've also worked on the UI (Java and JavaScript), the Database (SQLite), installers and web server. Tools: Dev Studio, Eclipse, bash, perl, Firebug, VirtualBox.

SENIOR COMPUTER SCIENTIST, ADOBE SYSTEMS, SAN JOSE, CALIFORNIA. 1998-2007

I joined the company in England as a Consulting Engineer (OEM support for PostScript).
In 2000 I was invited to relocate to San Jose, CA as a Project Lead Engineer in the Print Technology Group. I really enjoyed leading a team of 14 to develop a web printing application.
I also spent several years in Core Technology working on ExtendScript - the company's JavaScript interpreter. I gave presentations at the Adobe Tech Summit (engineering conference). Almost all code at Adobe is developed for Windows and Apple (OS-X today; OS 9 in the early days) and often Linux.

DIRECTOR, ROBIN MILLS SOFTWARE CONSULTANCY

I had my own business for 4 years and had a very successful 3 year contract with AGFA in Antwerp, Belgium to develop AgfaScript - a cross-platform PostScript interpreter (Solaris, AIX and Windows)

OPEN SOURCE AND OTHER ACTIVITY

I contribute to three open-source projects involving exif photograhic meta data: exiv2 (C++ exif library), pyexiv2 (python wrapper for exiv2) and Phatch for which I developed the GeoTag Action.
My builds (available from clanmills.com) of pyexiv2 and exiv2 for Windows and MacOS-X are unique.

Webmaster for a couple of clubs. Frequent contributor to forums.macosxhints.com. Moderator on forum.whatismyip.com (a network forum). Fit & healthy and qualified/ran in the Boston Marathon in 2006. Happily married. European Citizen and US resident (green card).

Skills

I've written and fixed a lot of code. I'm quick at learning new technologies and reading and understanding other people's code. I rank my skill set at:

Very	C++ and JavaScript	Dev Studio, XCode
Strong	Windows, MacOS-X	Presentations, Training Courses, Public Speaking
	Build Engineering	Prototyping and demonstrations
Strong	Scripting	Web technologies (HTML, CSS, HTTP, CGI, ASP)
	Linux	Perl and Python
	Writing documentation	Eclipse
Good	Java	Database

Education

University of Glasgow, Scotland. B Sc with First Class Honours in Engineering.

Referrals (email/phone numbers provided on request)

Dennis Connor	Director of Development, Novariant Incorporated, Fremont CA.
Richard Dermer	Senior Computer Scientist, Adobe Systems Incorporated, Seattle WA.
Jim Poje	Senior Computer Scientist, Adobe Systems Incorporated, San Jose CA.

Robin Mills
January 15, 2010

TOUGHEST ASSIGNMENT

This is easy! The PostScript Interpeter at AGFA was a challenge and the garbage collector was the most difficult part of the job.

I was with AGFA for 3 years. The task was to convert their PostScript Level 1 interpreter to PostScript Level 2 and to ship it on a variety of printing devices. The job took 3 years and I contributed 70,000 line of portable C code which ran on Solaris (Sparc and x86), IBM/AIX PPC, DOS and Windows NT 3.5. It supported little-endian and big-endian 32 bit architectures. My role in the project was to write the PostScript Language Interpreter and other engineers dealt with graphics, color, screening, fonts, and product integration with hardware. Every byte that enters an AGFA printing product runs the gauntlet of my code!

I tackled it in three stages:

1) I started adding PostScript Level 2 capability while trying to understand the code that was already written. This stage continued for about 9 months before I came to the conclusion that it would be quicker to rewrite the existing code than to keep patching and fixing it.

2) I rewrote the main body of the interpreter. Within about 3 months I was ahead of where we were at the start of the rewrite and moving much more quickly. In less than a year we were approaching code complete. I wrote around 50,000 lines of the code in less than a year.

3) Test and Integration took about 1 year. We purchased a third-party test suite for PostScript and dealt with every differences between Adobe's implementation and our interpreter. I also polished and optimized the code for performance. The interpreter was about 2x the performance of the Adobe implementation.

The most difficult part was the Garbage Collector. This uses a recursive mark and sweep technique. There's no magic to getting it to work. The design is very simple and the debugging requires a cool and sober head. The garbage collector is often 100 levels deep in recursion.

Things that helped hugely in this task were:

• Good System Architecture Drawings with which to discuss issues with the team

• Liberal Use of ASSERT (and other configurable Macros)

• Extensive debugging instrumentation which could be called from PostScript or from dbxtool

• Very good test suite

• Great Team Leader

Robin Mills

```c
#include <stdio.h>
#include <stdlib.h>

int syntax() { printf("syntax: lister arg+\n") ; return 1 ; }

/* singly linked list data structure */
typedef struct List_s {
    const char*     data ;
    struct List_s*  next ;
} List_t, *List_p,**List_h;

List_p List_reverse(List_h h)
{
    List_p p = *h;
    if   ( p ) {
        List_p prev = NULL;
        while (p)
        {
            List_p temp = p->next;
            p->next     = prev   ;
            prev        = p      ;
            p           = temp   ;
        }
        *h = prev;
    }
}

void List_print(const char* msg,List_p p)
{
    if ( p ) {
        printf("%s: ",msg) ;
        List_p node = p ;
        while (node) {
            printf("%s ",node->data) ;
            node = node->next ;
        }
        printf("\n") ;
    }
}

void List_free(List_h h)
{
    List_p p = *h ;
    if  (  p ) {
        List_p node = p ;
        while (node) {
            void* temp = node ;
            node = node->next ;
            free(temp) ;
        }
        *h = NULL ;
    }
}

int main(int argc, const char* argv[])
{
    int i ;
    List_p list = NULL ;
    /* create a list from argv */
    for ( i = 1 ; i < argc ; i++ ) {
        List_p node = (List_p) malloc(sizeof(List_t)) ;
        if ( node ) {
            node->data  = argv[argc-i] ;
            node->next  = NULL ;
            if ( list ) {
                node->next = list ;
            }
            list = node ;
        }
    }

    List_print  ("forward: " ,list) ;
    List_reverse(&list) ;
    List_print  ("reverse: " ,list) ;
    List_free   (&list) ;

    return argc < 2 ? syntax() : 0 ;
}
```

```c
#include <stdio.h>
#include <string.h>
#include <stdlib.h>

/* ---------------------------------------------------------- */
/* normalize - convert DOS,UNIX,MAC line endings to UNIX       */
/* DOS = \r\n  MAC = \r  UNIX = \n                             */
/* ---------------------------------------------------------- */
void normalize(char* s)
{
    int   r = 0 ; // read position
    int   w = 0 ; // write position
    for ( r = 0 ; s[r] ; r++ ) {
        char c =  s[r] ;
        if ( c == '\r' )
            if ( s[r+1] == '\n')
                r++ ; // ignore \n following \r
        if ( c == '\r' )
            c = '\n' ; // convert every \r to \n
        s[w++] = c ;
    }
    s[w] = 0 ;
}

char* a(char c) /* return ascii version of a char */
{
    static char buffer[10] ;
    sprintf(buffer,(32 <= c && c <= 127) ? "%c" : " %#02x " ,c);
    return buffer ;
}

int main(int argc,const char* argv[])
{
    int   i ;

    const char* S = "Apple\r\nBanana\r\nCarrot\rbanana\npear\r\napple\rstrawberry" ;
    char        s[256]  ;
    strcpy(s,S) ;

    normalize(s) ;

    for ( i = 0 ; i < strlen(s) ; i++ )
        printf("%s",a(s[i])) ;
    printf("\n");

    return 0 ;
}
```

```c
#include <stdio.h>
#include <stdlib.h>

size_t undup(int a[],size_t l)
{
    size_t w = 0 ; // write position
    size_t r = 0 ; // read  position
    while ( r < l ) {
        if ( a[w] != a[r++]  )
            a[++w] = a[r-1] ;
    }
    return w ;
}

int main(int argc,char* argv[])
{
    int A[] = {1,1,2,2,2,2,3,4,5,6,7,8,8,0};
    int a[100];
    size_t l;
    size_t i;
    for ( i = 0 ; A[i] ;i++ )
        a[i] = A[i] ;
    l = i+1 ;

    l = undup(a,l) ;

    for ( i = 0 ; i < l ; i++ )
        printf("%d ",a[i]);
    if ( l ) printf("\n") ;

    return 0 ;
}
```