Connectathon ‘98

64-Bit RPC

Alex Chiu

3/9/98

Why 64-bit?

- **Performance**
  - **64-bit UltraSPARC: 64-bit registers and extended instructions set**
- **Increased virtaul address space and file sizes**
  - **virtual address space: 18 EB (18×10^{18} bytes) per application**
  - **file size: up to 9 EB**
- **64-bit applications**
  - **Data Warehousing, Multimedia, ...**
- **Other vendors**
  - **Digital, SGI, HAL -- already provide a 64-bit environment**
  - **HP, SCO, IBM, Microsoft -- 64-bit roadmaps**

Data Model

- **ILP32 - C data type model for 32-bit Solaris**
- **LP64 - C data type model for 64-bit Solaris**
- **Comparison**

Data Type	ILP32(bits)	LP64(bits)
char	8	8
short	16	16
int	32	32
long long	64	64
long	32	64
pointer	32	64

64-bit Solaris

- **Kernel and apps support requirements**

Kernel	Application	Support
32-bit	32-bit	Yes
32-bit	64-bit	No
64-bit	32-bit	Yes
64-bit	64-bit	Yes

Binary compatibility: 32-bit applications must run on 64-bit Solaris without needing a recompile

- **Two sets of libraries generated from same source**
  - **32 bit applications only linked to 32-bit libraries (/usr/lib)**
  - **64 bit applications only linked to 64-bit libraries (/usr/lib/sparcv9)**

64-bit Solaris (Cont)

- **Commands and utilities remain 32-bit**
- **Debugging**
  - **32-bit debuggers can debug 32-bit programs.**
  - **64-bit debuggers can debug 32-bit or 64-bit programs**
- **Support of existing APIs**

Coding Practice

- **Don’t assume ints, longs, and pointers are same size**
- **Use proper format for printf() and scanf()**

```
main ()
{
unsigned long addr;
char *p;
      printf(“a 0x%x\n”, addr);
      printf(“p 0x%x\n”, p);
}
```

Fix: use 0x%lx for longs and %p for pointers

- **sizeof() returns an unsigned long**
- **Explicitly type cast to avoid type mismatch**

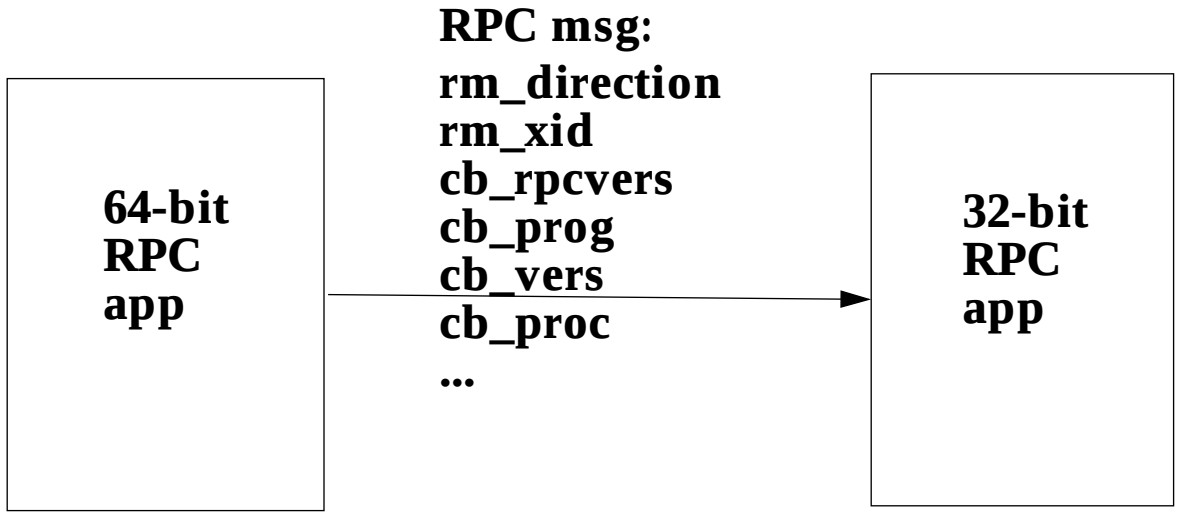
Coding Practice (Cont)

- **Use fixed size types such as int32_t over the wire**
- **Use derived types whenever appropriate**
- **Use pointer arithmetic instead of address arithmetic**
- **Big types**
  - **size_t, dev_t, time_t**

64-bit Safe RPC

- **Interoperability problem**

CLIENT *clnt_create(const char *host, *u_long prognum*, *u_long versnum*, **const char *nettype)**



64-bit Safe RPC (Cont)

- **A solution**

```
#if defined(_LP64) || defined(_I32LPx)
        typedef uint32_t rpcprog_t;
        typedef uint32_t rpcvers_t;
        typedef uint32_t rpcproc_t;
#else
        typedef unsigned long rpcprog_t;
        typedef unsigned long rpcvers_t;
        typedef unsigned long rpcproc_t;
#endif
```

CLIENT *clnt_create(const char *host, *rpcprog_t prognum*, ***rpcvers_t*** *versnum*, **const char *nettype)**

64-bit Safe RPC (Cont)

- **xdr_long()**

```
bool_t
xdr_long(XDR *xdrs, long *lp)
{
        bool_t          dummy;
        int32_t         i;

        if (xdrs->x_op == XDR_ENCODE) {
#if defined (_LP64)
            if ((*lp > INT32_MAX) || (*lp < INT32_MIN)) {
                (void) syslog(LOG_ERR, xdrlong_err, (const char *) "long");
                return (FALSE);
            }
```

64-bit Safe RPC (Cont)

```
#endif
            i = (int32_t)*lp;
            dummy = XDR_PUTINT32(xdrs, &i);
        } else if (xdrs->x_op == XDR_DECODE) {
            dummy = XDR_GETINT32(xdrs, &i);
            *lp = (long)i;
        } else if (xdrs->x_op == XDR_FREE)
            dummy = TRUE;
        else
            dummy = FALSE;
        return (dummy);
}
```

- **xdr_u_long()**

64-bit Safe RPC (Cont)

- **Kernel RPC**
  - **no reference to xdr_long() and friends**
  - **no reference to IXDR_PUT_LONG() and friends**
- **User RPC**
  - **xdr_long() and friends kept for backward compatibility**
  - **IXDR_PUT_LONG() and friends kept for backward compatibility**
- **rpcgen**
  - **modified to ensure source backward compatibility**
  - **Solaris 2.6 rpcgen'ed source files continue to compile and run on 2.7 machines as 32-bit apps in the 64-bit environment**
  - **Solaris 2.7 rpcgen'ed source files compile and run on 2.6 machines**

System Calls

- **On 32-bit kernel, “native” system calls are 32-bit**
- **On 64-bit kernel, “native” system calls are 64-bit, and “compatible” system calls are 32-bit**
- **NFS system calls - nfssys() and mount() use both**

System Calls (Cont)

```
/* Native data structure */

struct exargs {
    caddr_t ptr;
    size_t len;
};

#ifdef _SYSCALL32

/* Kernel view of 32-bit data structure */

struct exargs32 {
    caddr32_t ptr;
    size32_t len;
};

#endif /* _SYSCALL32 */
```

System Calls (Cont)

```
int
example(int arg, void *p)
{
    struct exargs;

    if (get_udatamodel() == DATAMODEL_NATIVE)
            copyin(p, &exargs, sizeof (exargs));
#ifdef _SYSCALL32_IMPL
    else {
            struct exargs32 exargs32;
            copyin(p, &exargs32, sizeof (exargs32));
            exargs.ptr = (void *)exargs32.ptr;
            exargs.len = exargs32.len;
    }
#endif

    /* common code from here on .. */
```