TINE CORE MEETING

26.4.2016

Noteworthy C-Lib Changes

- Bug fix concerning **poll**() behavior.
- **MCA** format elevation
- Bug fix on **MACOS** (*from last time*)

select() vs. poll()

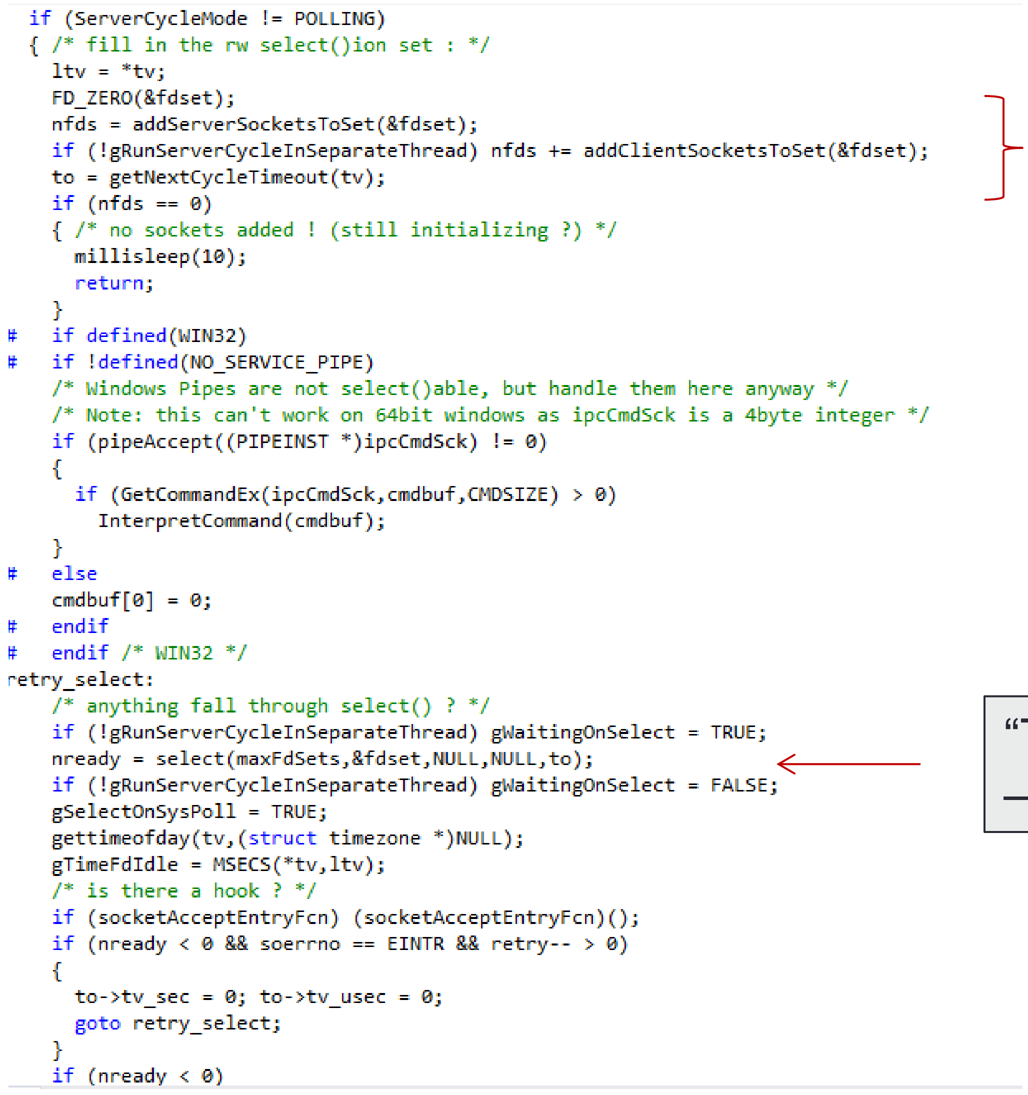
- Up to August 2015 TINE made use of *select*().

```
int select(int nfds, fd_set *readfds, fd_set *writefds,
                fd_set *exceptfds, struct timeval *timeout);

    void FD_CLR(int fd, fd_set *set);
    int  FD_ISSET(int fd, fd_set *set);
    void FD_SET(int fd, fd_set *set);
    void FD_ZERO(fd_set *set);
```

```
RETURN VALUE
    On  success,  select() and pselect() return the number of file descriptors contained in the three returned descriptor
sets (that is, the total number of bits that  are set in readfds, writefds, exceptfds) which may be zero if the timeout expires
before anything interesting happens.  On error,  -1  is  returned,  and  errno  is  set  appropriately; the sets and timeout
become undefined, so do not rely on their contents after an error.

ERRORS
    EBADF  An invalid file descriptor was given in one of  the  sets.   (Perhaps  a
         file  descriptor  that  was already closed, or one on which an error has
         occurred.)

    EINTR  A signal was caught; see signal(7).

    EINVAL nfds is negative or the value contained within timeout is invalid.

    ENOMEM unable to allocate memory for internal tables.
```

select() vs. poll()

in routine AcceptIP() in srvip.c



```
 if (ServerCycleMode != POLLING)
 { /* fill in the rw select()ion set : */
   ltv = *tv;
   FD_ZERO(&fdset);
   nfds = addServerSocketsToSet(&fdset);
   if (!gRunServerCycleInSeparateThread) nfds += addClientSocketsToSet(&fdset);
   to = getNextCycleTimeout(tv);
   if (nfds == 0)
   { /* no sockets added ! (still initializing ?) */
     millisleep(10);
     return;
   }
#  if defined(WIN32)
#  if !defined(NO_SERVICE_PIPE)
   /* Windows Pipes are not select()able, but handle them here anyway */
   /* Note: this can't work on 64bit windows as ipcCmdSck is a 4byte integer */
   if (pipeAccept((PIPEINST *)ipcCmdSck) != 0)
   {
     if (GetCommandEx(ipcCmdSck,cmdbuf,CMDSIZE) > 0)
       InterpretCommand(cmdbuf);
   }
#  else
   cmdbuf[0] = 0;
#  endif
#  endif /* WIN32 */
retry_select:
   /* anything fall through select() ? */
   if (!gRunServerCycleInSeparateThread) gWaitingOnSelect = TRUE;
   nready = select(maxFdSets,&fdset,NULL,NULL,to);
   if (!gRunServerCycleInSeparateThread) gWaitingOnSelect = FALSE;
   gSelectOnSysPoll = TRUE;
   gettimeofday(tv,(struct timezone *)NULL);
   gTimeFdIdle = MSECS(*tv,ltv);
   /* is there a hook ? */
   if (socketAcceptEntryFcn) (socketAcceptEntryFcn)();
   if (nready < 0 && soerrno == EINTR && retry-- > 0)
   {
     to->tv_sec = 0; to->tv_usec = 0;
     goto retry_select;
   }
   if (nready < 0)
```

put all socket descriptors in the set

just select on ‘read sets’ !

“The mother of all select()s” – S. Herb

The doocs server with >1024 open files problem :

- **XFEL** Magnet Middle Layer Server (L. Froehlich) written as a doocs server with TINE thread.
- Collects info from all 1200+ Magnet PSCs in **XFEL**.
- Keeps a doocs history of each one individually.
  - doocs histories keep files open!
- Then turns on the TINE thread
- Initial socket descriptor tries to begin at > 1200.
- *select*() is limited to 1024
- Oops!

Solutions …

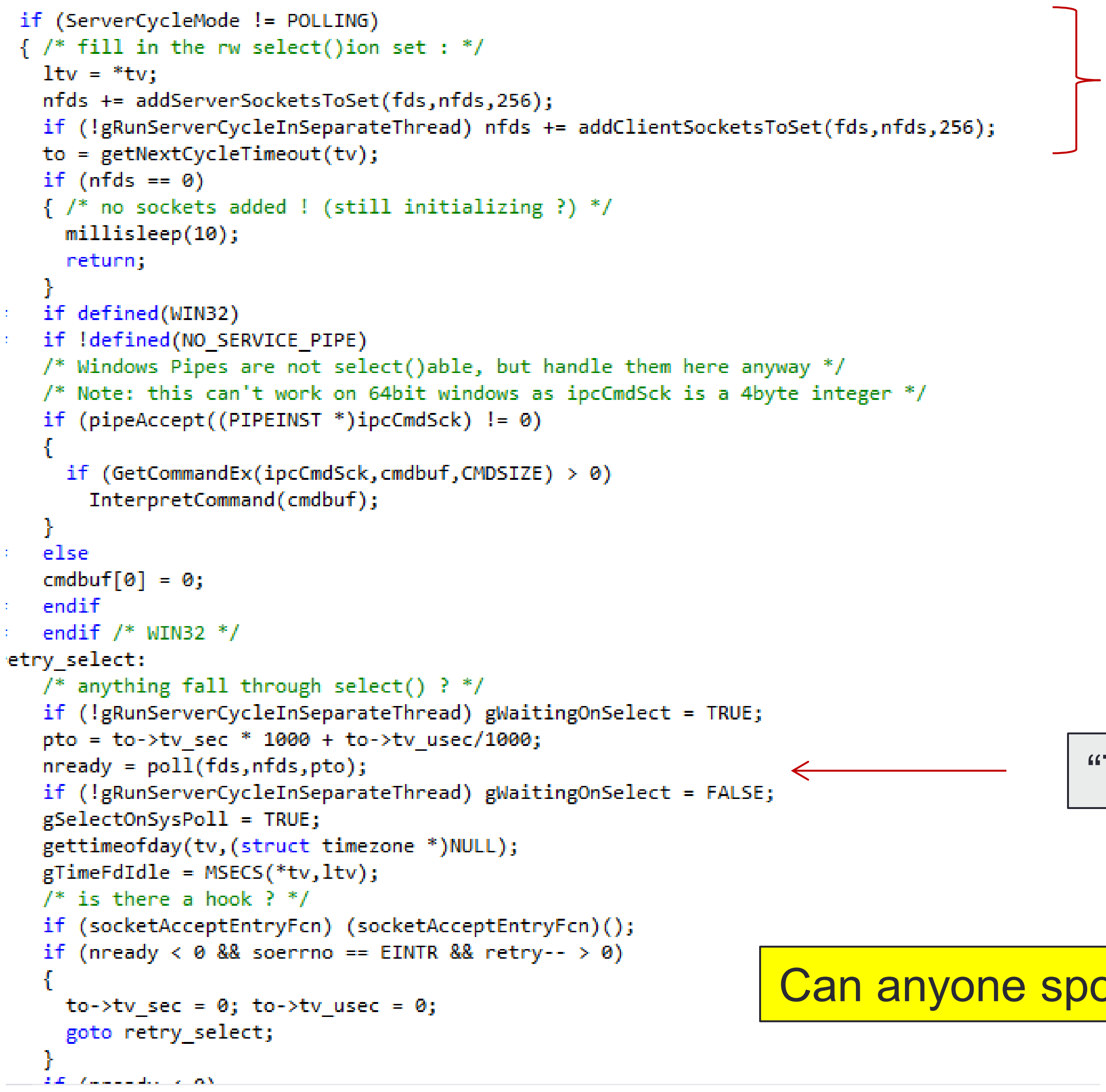
- Don't do it this way (?).
- Switch to the more modern '*poll*()' call.
  - *select*() is a BSD standard since the 70s and 80s.
  - *poll*() does not exist on all platforms: missing on e.g.Windows XP and VxWorks

```
int poll(struct pollfd *fds, nfds_t nfds, int timeout);

struct pollfd {
        int   fd;         /* file descriptor */
        short events;     /* requested events */
        short revents;    /* returned events */
    };
```

```
POLLIN
       There is data to read.
POLLPRI
       There is urgent data to  read  (e.g.,  out-of-band  data  on  TCP
        socket;  pseudoterminal  master  in  packet  mode  has seen state
        change in slave).
POLLOUT
       Writing now will not block.
POLLRDHUP (since Linux 2.6.17)
        Stream socket peer closed connection, or shut down  writing  half
        of  connection.   The  _GNU_SOURCE  feature  test  macro  must be
        defined (before including any header files) in  order  to  obtain
        this definition.
POLLERR
        Error condition (output only).
POLLHUP
        Hang up (output only).
```

```
RETURN VALUE
     On  success,  a  positive  number is returned; this is the
number of structures which have nonzero revents fields  (in
other  words,  those  descriptors  with events or errors
reported).  A value of 0 indicates that the call timed out and
no file descriptors were ready.  On error, -1 is returned,  and
errno  is  set  appropriately.

ERRORS
     EFAULT The  array  given as argument was not
contained in the calling program's address space.

     EINTR  A signal occurred before any requested event;
see signal(7).

     EINVAL The nfds value exceeds the RLIMIT_NOFILE
value.

     ENOMEM There was no space to allocate file
descriptor tables.
```

Solution with poll() in August 2015



```
 if (ServerCycleMode != POLLING)
 { /* fill in the rw select()ion set : */
   ltv = *tv;
   nfds += addServerSocketsToSet(fds,nfds,256);
   if (!gRunServerCycleInSeparateThread) nfds += addClientSocketsToSet(fds,nfds,256);
   to = getNextCycleTimeout(tv);
   if (nfds == 0)
   { /* no sockets added ! (still initializing ?) */
     millisleep(10);
     return;
   }
   if defined(WIN32)
   if !defined(NO_SERVICE_PIPE)
   /* Windows Pipes are not select()able, but handle them here anyway */
   /* Note: this can't work on 64bit windows as ipcCmdSck is a 4byte integer */
   if (pipeAccept((PIPEINST *)ipcCmdSck) != 0)
   {
     if (GetCommandEx(ipcCmdSck,cmdbuf,CMDSIZE) > 0)
       InterpretCommand(cmdbuf);
   }
   else
   cmdbuf[0] = 0;
   endif
   endif /* WIN32 */
etry_select:
   /* anything fall through select() ? */
   if (!gRunServerCycleInSeparateThread) gWaitingOnSelect = TRUE;
   pto = to->tv_sec * 1000 + to->tv_usec/1000;
   nready = poll(fds,nfds,pto);
   if (!gRunServerCycleInSeparateThread) gWaitingOnSelect = FALSE;
   gSelectOnSysPoll = TRUE;
   gettimeofday(tv,(struct timezone *)NULL);
   gTimeFdIdle = MSECS(*tv,ltv);
   /* is there a hook ? */
   if (socketAcceptEntryFcn) (socketAcceptEntryFcn)();
   if (nready < 0 && soerrno == EINTR && retry-- > 0)
   {
     to->tv_sec = 0; to->tv_usec = 0;
     goto retry_select;
   }
   if (nready < 0)
```

put all socket descriptors in the set

just look for POLLIN !

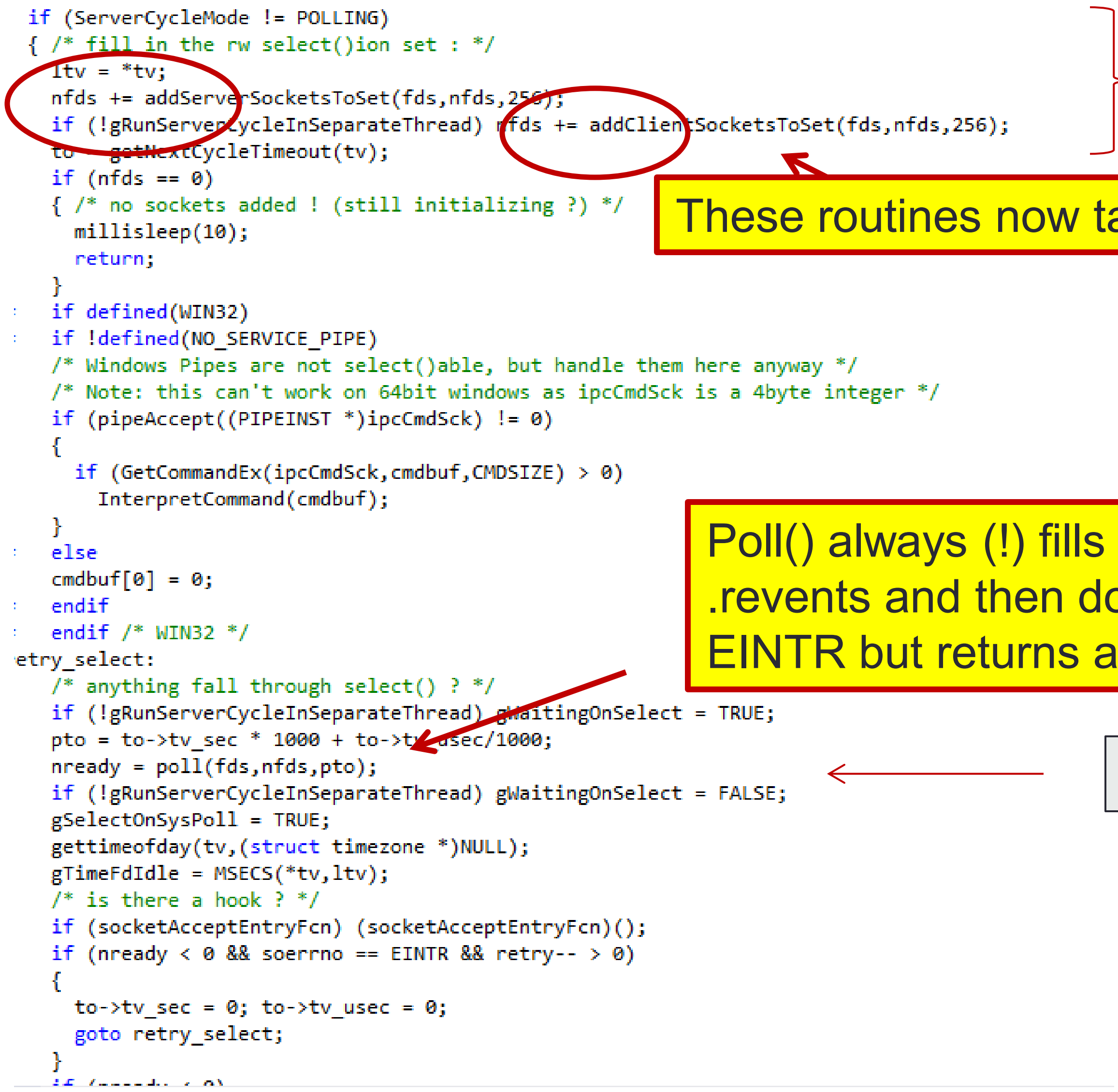
“The mother of all poll()s”

Can anyone spot the bugs ?

Recent Problems …

- CFEL:
  - Using TCP, client disconnects and server consumes 100% for up to 5 minutes.
- EMBL:
  - tineRepeater occasionally gets into a mode where it uses 100% CPU (and stays there until restarted).
- ????

Bugs with poll() since August 2015



```
 if (ServerCycleMode != POLLING)
 { /* fill in the rw select()ion set : */
   ltv = *tv;
   nfds += addServerSocketsToSet(fds,nfds,256);
   if (!gRunServerCycleInSeparateThread) nfds += addClientSocketsToSet(fds,nfds,256);
   to = getNextCycleTimeout(tv);
   if (nfds == 0)
   { /* no sockets added ! (still initializing ?) */
     millisleep(10);
     return;
   }
:  if defined(WIN32)
:  if !defined(NO_SERVICE_PIPE)
   /* Windows Pipes are not select()able, but handle them here anyway */
   /* Note: this can't work on 64bit windows as ipcCmdSck is a 4byte integer */
   if (pipeAccept((PIPEINST *)ipcCmdSck) != 0)
   {
     if (GetCommandEx(ipcCmdSck,cmdbuf,CMDSIZE) > 0)
       InterpretCommand(cmdbuf);
   }
:  else
   cmdbuf[0] = 0;
:  endif
:  endif /* WIN32 */
etry_select:
   /* anything fall through select() ? */
   if (!gRunServerCycleInSeparateThread) gWaitingOnSelect = TRUE;
   pto = to->tv_sec * 1000 + to->tv_usec/1000;
   nready = poll(fds,nfds,pto);
   if (!gRunServerCycleInSeparateThread) gWaitingOnSelect = FALSE;
   gSelectOnSysPoll = TRUE;
   gettimeofday(tv,(struct timezone *)NULL);
   gTimeFdIdle = MSECS(*tv,ltv);
   /* is there a hook ? */
   if (socketAcceptEntryFcn) (socketAcceptEntryFcn)();
   if (nready < 0 && soerrno == EINTR && retry-- > 0)
   {
     to->tv_sec = 0; to->tv_usec = 0;
     goto retry_select;
   }
   if (nready < 0)
```

put all socket descriptors in the set

These routines now take an offset !

just look for POLLIN !

Poll() always (!) fills in the exceptions in .revents and then does NOT return -1 with EINTR but returns a ready count > 0!

“The mother of all poll()s”

MCA Format Elevation

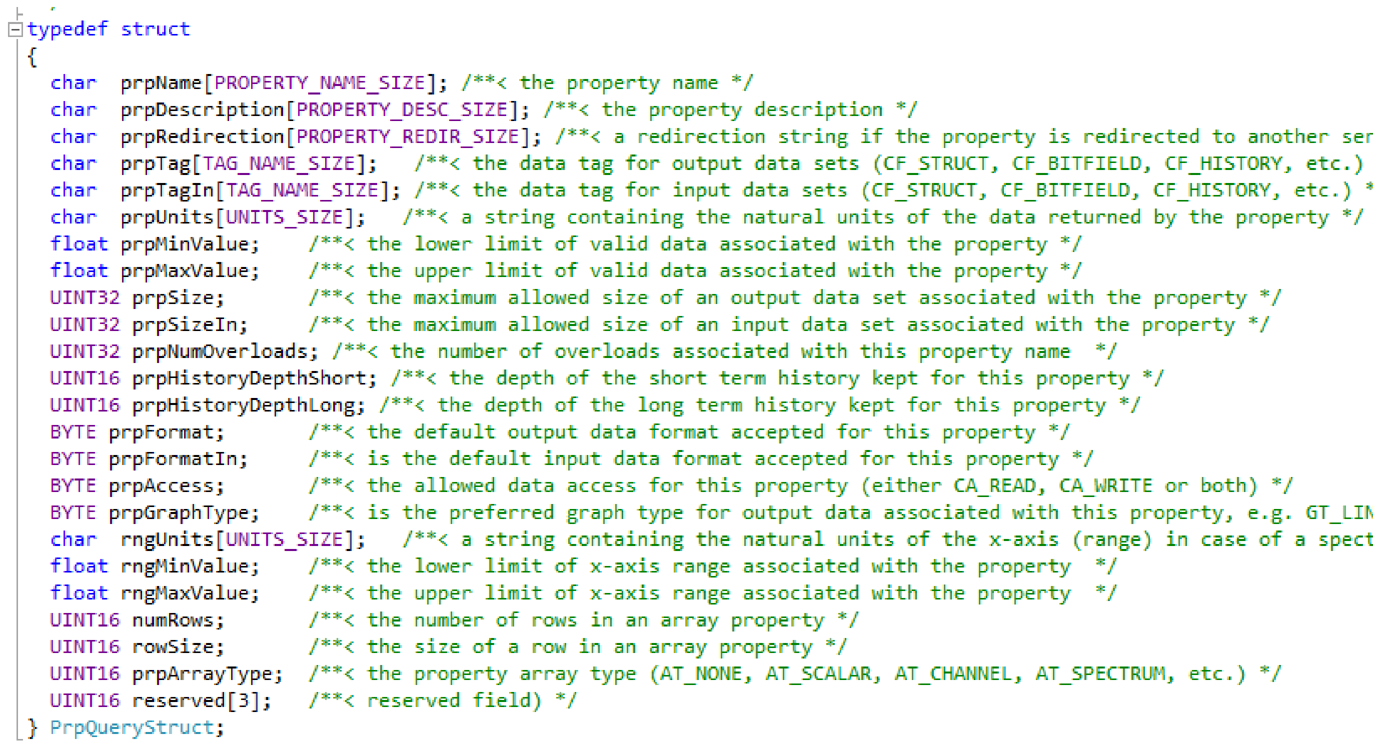
- A property can register itself to deliver a '*multi-channel array*' (**MCA**).
  - e.g. property "**Orbit.X**" (all BPM positions), "**Pressure**" (all ion pumps), "**Temperature**" (all temperature sensors)
- More efficient to get an array of *all 300 monitors* than to get *300 individual elements* !
- A "proper" **MCA** property will be able to deliver
  - a value for a single requested device, (scalar)
  - a set of values for a section, or (array)
  - all values for all devices. (array)
- Monitoring a single device will '*coerce*' the requested contract into obtaining the entire array and piping the proper array element into the original request.
  - *The server sees and handles only a single contract for all devices on behalf of ALL interested parties*.

MCA Format Elevation

- The Issue:
  - What if device ‘B’ (*e.g. element #15*) has a problem (*e.g. ‘hardware error’*)?
  - Either the whole **MCA** has a problem (‘*hardware error*’) or the whole **MCA** is ‘*okay*’ (*and the data for element #15 ?*).
    - Josef’s server wizard takes the first strategy.
- New Approach:
  - Format elevation !
  - If a property “P” is registered to deliver an **MCA** of **FLOAT** values AND is overloaded to deliver an **MCA** of **FLTINT** values (value-status pairs) then the **MCA** coercion is allowed to ‘elevate’ a request for **FLOAT**s to a request for **FLTINT**s.
  - The original request sees the **FLOAT** value and the **INT** status.

MACOS Bug Fix (from last time)

- MACOS (FreeBSD) doesn’t like in-situ memcpy() (overlapping memory areas).
- e.g. Prepare an array of ‘PrpQueryStruct’ items



```
typedef struct
{
  char  prpName[PROPERTY_NAME_SIZE]; /**< the property name */
  char  prpDescription[PROPERTY_DESC_SIZE]; /**< the property description */
  char  prpRedirection[PROPERTY_REDIR_SIZE]; /**< a redirection string if the property is redirected to another ser
  char  prpTag[TAG_NAME_SIZE];   /**< the data tag for output data sets (CF_STRUCT, CF_BITFIELD, CF_HISTORY, etc.)
  char  prpTagIn[TAG_NAME_SIZE]; /**< the data tag for input data sets (CF_STRUCT, CF_BITFIELD, CF_HISTORY, etc.) *
  char  prpUnits[UNITS_SIZE];   /**< a string containing the natural units of the data returned by the property */
  float prpMinValue;    /**< the lower limit of valid data associated with the property */
  float prpMaxValue;    /**< the upper limit of valid data associated with the property */
  UINT32 prpSize;       /**< the maximum allowed size of an output data set associated with the property */
  UINT32 prpSizeIn;     /**< the maximum allowed size of an input data set associated with the property */
  UINT32 prpNumOverloads; /**< the number of overloads associated with this property name  */
  UINT16 prpHistoryDepthShort; /**< the depth of the short term history kept for this property */
  UINT16 prpHistoryDepthLong; /**< the depth of the long term history kept for this property */
  BYTE  prpFormat;      /**< the default output data format accepted for this property */
  BYTE  prpFormatIn;    /**< is the default input data format accepted for this property */
  BYTE  prpAccess;      /**< the allowed data access for this property (either CA_READ, CA_WRITE or both) */
  BYTE  prpGraphType;   /**< is the preferred graph type for output data associated with this property, e.g. GT_LIN
  char  rngUnits[UNITS_SIZE];   /**< a string containing the natural units of the x-axis (range) in case of a spect
  float rngMinValue;    /**< the lower limit of x-axis range associated with the property  */
  float rngMaxValue;    /**< the upper limit of x-axis range associated with the property  */
  UINT16 numRows;       /**< the number of rows in an array property */
  UINT16 rowSize;       /**< the size of a row in an array property */
  UINT16 prpArrayType;  /**< the property array type (AT_NONE, AT_SCALAR, AT_CHANNEL, AT_SPECTRUM, etc.) */
  UINT16 reserved[3];   /**< reserved field) */
} PrpQueryStruct;
```

MACOS Bug Fix (from last time)

- Caller just wants an array of **NAME64** items (64-character fixed length strings).

```
for (i=0; i<length; i++) memcpy(names[i],prpQueryStructs[i].prpName,64);
```

- Should work just fine if ‘*names*’ and ‘*prpQueryStructs*’ point to the same memory area …

- MACOS doesn’t like this …