Tip of the Week :

- using user debug functions with attachfec

Does not apply to java servers (sorry!)

- Two ways to register command-line interface routines:
  - RegisterUserCommand(cmd,fcn, iparam, fparam, access)
    - Cmd is the

Does not apply to java servers (sorry!)

```
int RegisterUserCommand ( char *                cmd,
                          int(*)(int, int, int, int)  fcn,
                          int *                 iparam,
                          float *               fparam,
                          int                   access
                        )
```

Registers a user-defined command or variable which can be accessed via the TINE command line interpreter.

As a console application, a running tine server (or client) offers a variety of services at the command line (just type 'help'). For instance, you can turn debugging on or off, get the current server statistics, examine the connection tables, etc. This interface is avaible via the 'remote' accessor even if the server is running in the background. It it often frequently desireable to examine or alter server-specific variables or call a server specific routine. This can be partially achieved by making use of this registration function.

Parameters:

cmd is the command string to be parsed by the command interpreter.
fcn is an optional user defined function taking four integer arguments.
iparam is a pointer to an optional global integer variable.
fparam is a pointer to an optional global float variable.
access is a TINE access code (either CA_READ or CA_READ|CA_WRITE). To allow 'set' commands at the command line, you should include the CA_WRITE flag.

Note: Only one of fcn, iparam, or fparm, should be non-NULL when making this call. If the users types "get <cmd>" without arguments at the command line, then either iparam or fparam is displayed on the console, depending on which parameter was non-NULL at registration time. Likewise "set <cmd> = <val>" can set this variable to a value entered at the console. If a function is registered, then "get <cmd>(arg1,arg2,arg3,arg4)" will call the function registered with the arguments supplied. "set <cmd> (arg1,arg2,arg3,arg4) = <val>" will set all arguments equal to <val> and then call the function. If fewer arguments than 4 are supplied then the function will be called with '0' for the missing arguments.

A more reasonable command line function registration call is underway. However the routine so presented is generally sufficient for checking hardware readouts etc. and checking and setting the contents of global variables.

Returns:

0 if successful, otherwise a TINE completion code

```
int RegisterUserFunction ( char *                  name,
                           int(*)(int argc, char *argv[])  fcn
                         )
```

Registers a user-defined function which can be accessed via the TINE command line interpreter.

As a console application, a running tine server (or client) offers a variety of services at the command line (just type 'help'). For instance, you can turn debugging on or off, get the current server statistics, examine the connection tables, etc. This interface is avaible via the 'remote' accessor even if the server is running in the background. It it often frequently desireable to examine or alter server-specific variables or call a server specific routine. This can be partially achieved by making use of this registration function.

Example 1: RegisterUserCommand()

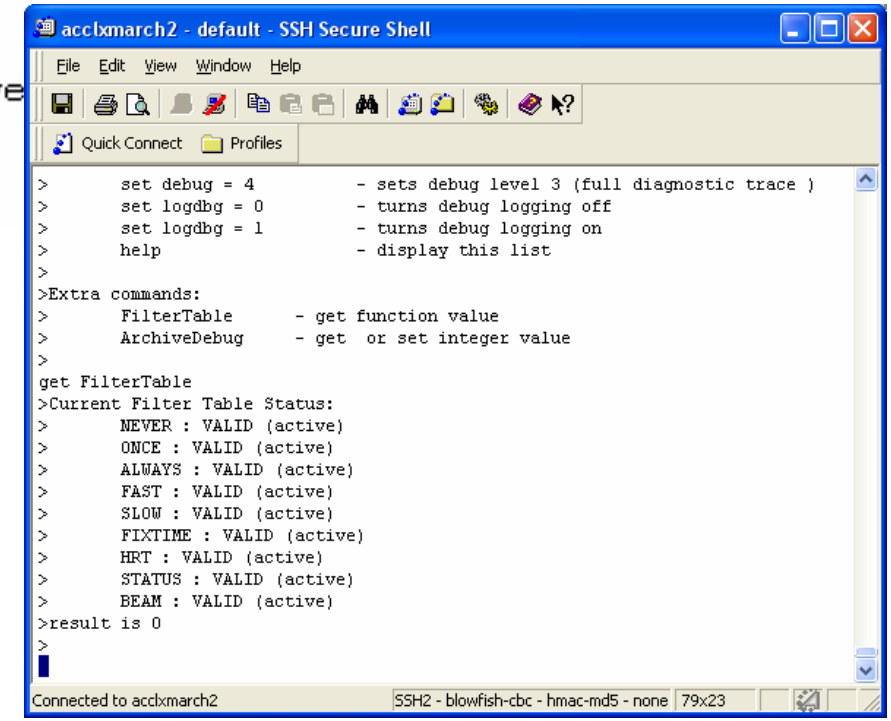
Initialization routine from the central archiver:

```
void mexinit(void)
{
  DTYPE dout;
  char *data=NULL;
  int i, j, k, cc = 0;

  RegisterDeviceName(MEX_EQMNAME,"keyword",0);

  remove("problems.log");
  RegisterUserCommand("ArchiveDebug",NULL,&archdebug,NULL,CA_READ+CA_WRITE);
  RegisterUserCommand("FilterTable",dumpFilterTable,NULL,NULL,CA_READ);
  RegisterUserCommand("ArchiveRecord",dumpArchiveRecord,NULL,NULL,CA_READ);

  daqInit(srvinfo.dbfile[0],srvinfo.ndbfiles);
  nredirfiles = getRedirInfo();
  redirInit(redirfile,redirsrv,nredirfiles);

  getNetGlobalList(GLOBALS_FILENAME);
  if (ngloballist > 0)
  { /* found a registered globals file */
    feclog("found %d globals from list %s",ngloballist,GLOBALS_FILENAME);
    for (i=0; i<nrecords; i++) for (j=0; j<daqTbl[i].nparam; j++)
      for (k=0; k<ngloballist; k++)
```

Example 1: continued

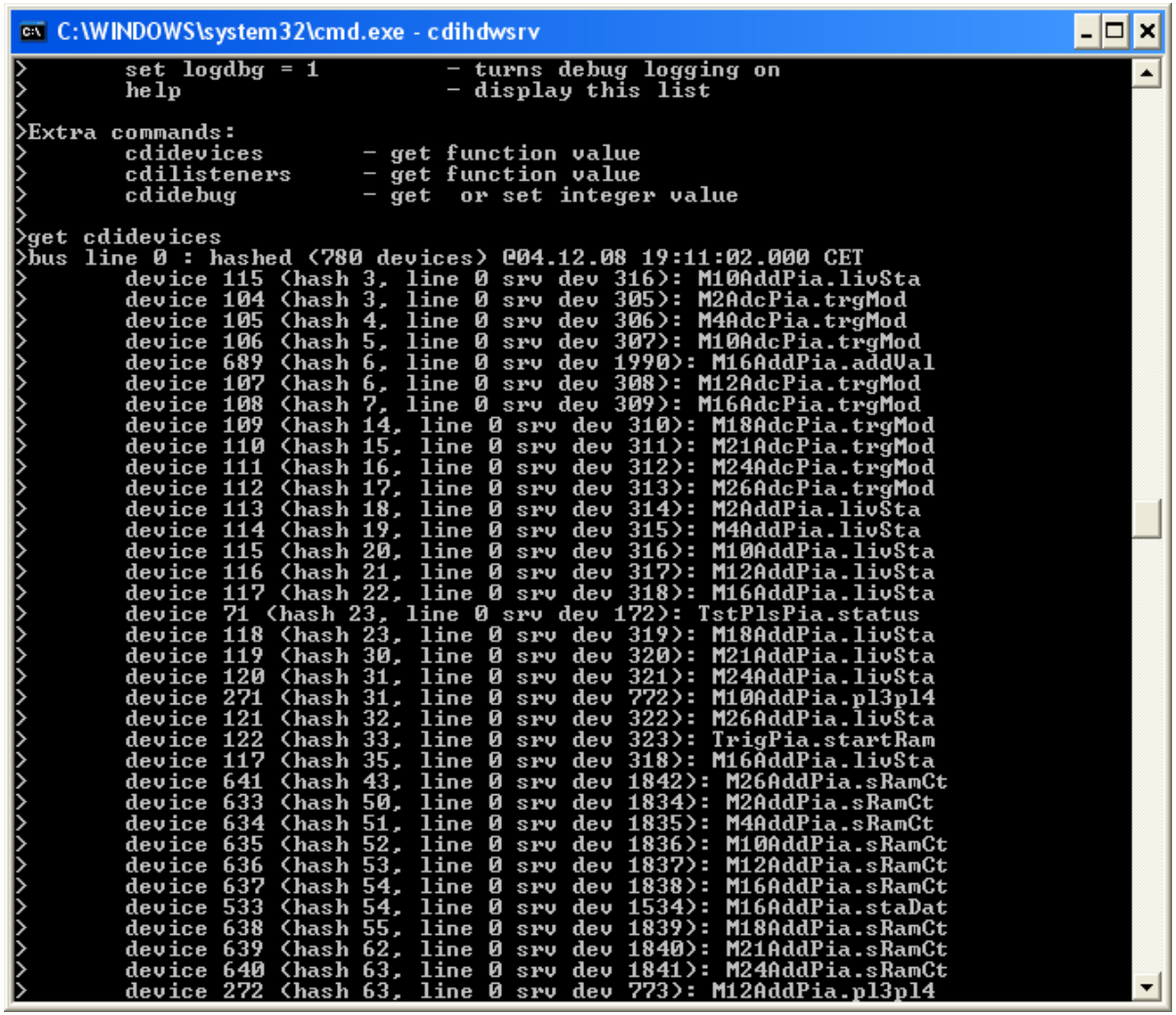
```
int dumpFilterTable(int a0,int a1,int a2,int a3)
{
  int i;
  a0 = a0; a1 = a1; a2 = a2; a3 = a3;
  ttyoutput("Current Filter Table Status:");
  for (i=0; i<nDaqFilterTblEntries; i++)
  {
    ttyoutput("\t%s : %s (%s)",
      daqFilterTbl[i].tag,
      daqFilterTbl[i].valid ? "VALID"
      daqFilterTbl[i].active ? "active
  }
  return 0;
}
```

Prints output onto stdout as well as any debug pipes!



Example 1: continued …

e.g. the CDI Hardware Server

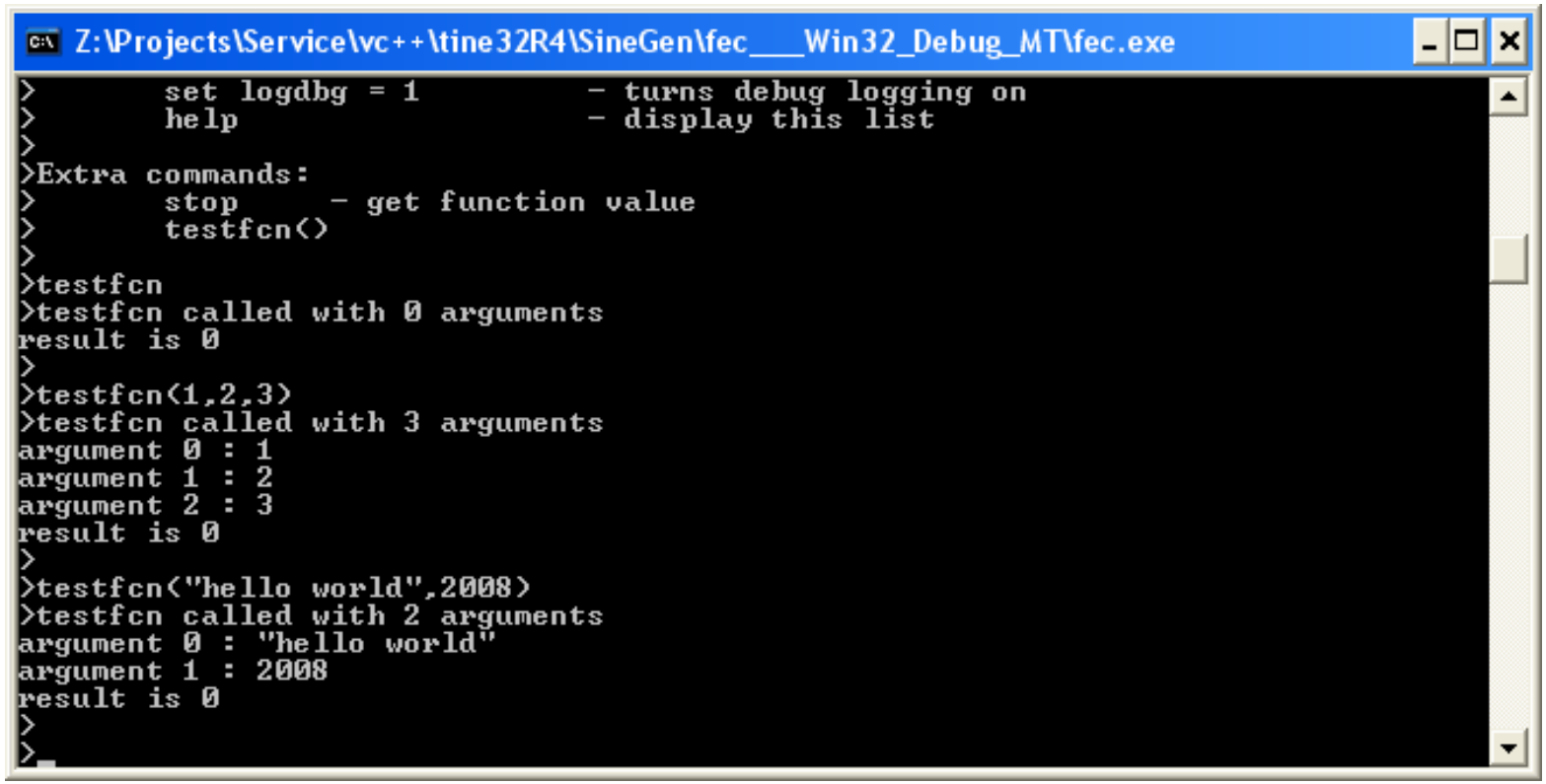


Example 2: RegisterUserFunction()

- Less ‘restrictive’, but you will have to ‘design more of it yourself’
- Prototype:
  - int RegisterUserFunction(char *name, int (*fcn)(int argc,char *argv[]))
- e.g.
  - RegisterUserFunction("testfcn",testfcn);

```
int testfcn(int argc,char *argv[])
{
  int i;
  printf("testfcn called with %d arguments\n",argc);
  for (i=0; i<argc; i++)
  {
    printf("argument %d : %s\n",i,argv[i]);
  }
  return 0;
}
```

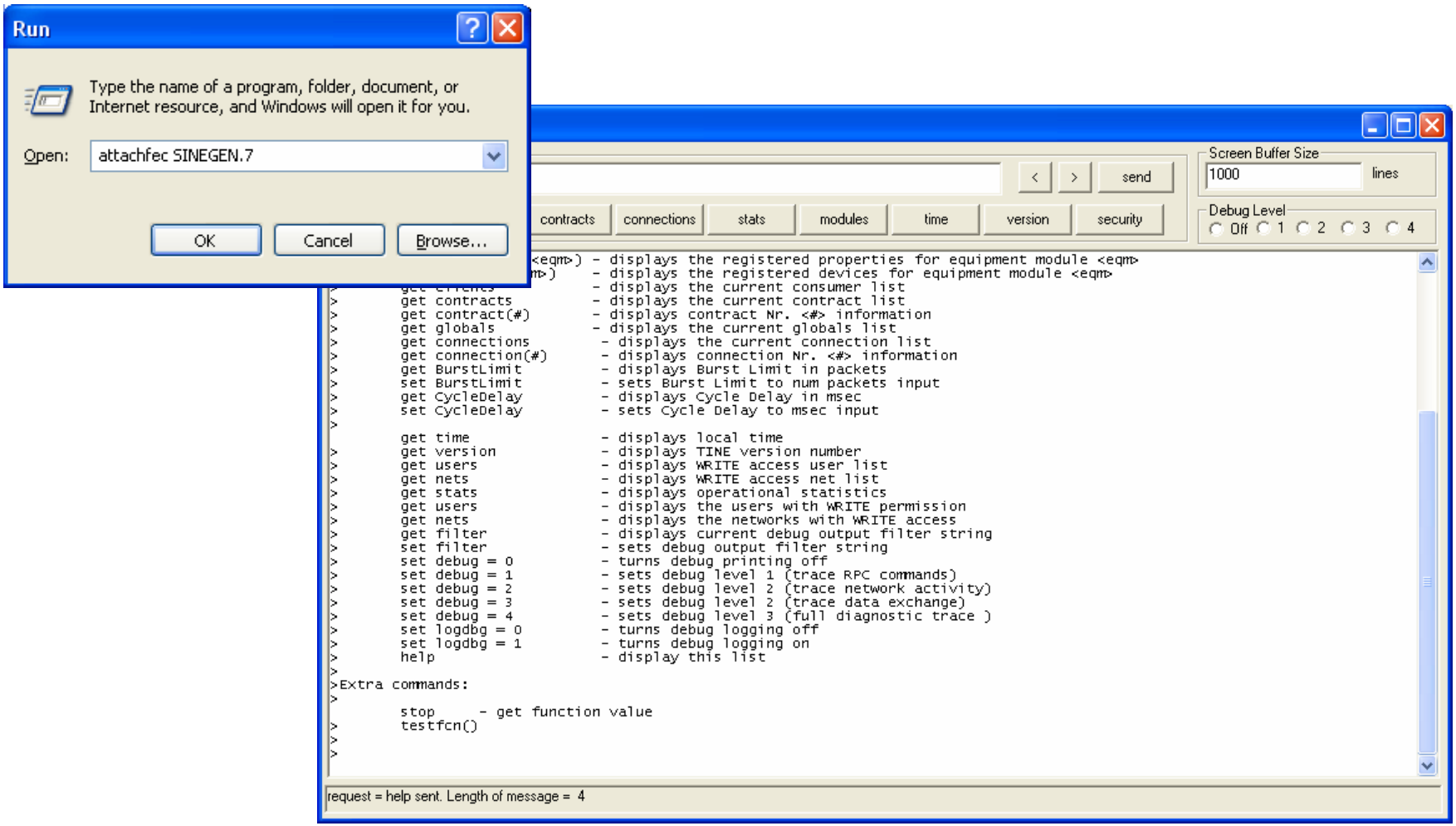
Example 2: continued



Getting there with Attachfec …

- If
  - server is a console program and is
  - running in the foreground and
  - you are sitting at the console
- Then you don’t need attachfec!
- Otherwise,
  - login to the machine running the server and type
  - “attachfec <fecname>”

Windows Example:



UNIX Example:

- ssh –l fecadmin acclxl2facil01
- attachfec L2CASFEC