Presentation Title: Cool Unix Utilties on MPE!
Long Presentation Title (70 chars): Using some cool Unix utilities on MPE to make life easier!

In this presentation you will see several examples of powerful Unix utility programs that are available on MPE/iX. You will see examples of combining MPE and POSIX commands at the MPE CI prompt and examples of using the power of the POSIX shell to combine MPE and POSIX command to accomplish complex tasks with tools freely available on MPE/iX.

There are several tools that are included with MPE/iX that came from the land of Unix and HP-UX. Most of these programs can be used as stand alone commands to operate on a file. The real power of these tools comes from the Unix concept of stringing a series of these tools together using pipes.

Some of these programs are simply relatives of their MPE cousins. For example:
"rm" vs. PURGE, and "mv" vs "RENAME". The advantage of the POSIX versions of these commands is that they actually work against the directory and can effective act on the file while it's being accessed. (Ok, to be honest, the equivalents are more like "rm" = "PURGELINK" and "mv" = "PURGEDIR", but I always forget about those…☹)

Other programs are entire languages like awk and perl. I will talk about how I like to use awk, but perl is best presented by itself.

There are several programs which are designed to be used in conjunction with others via pipes. Stringing together a sequence of commands an these "filter" type programs can make a complex task which would usually take a programmer (or JCL wizard) pages of code as simple as 1 or two lines of commands piped to one another.

Ok, so you're skeptical. Well, I was too. Let's do a quick example:

I need to find all the jobs logged on to the SYS account. The following command in the shell (sh.hpbin.sys) looks like this:

```
shell/iX> callci "showjob;job=@.sys"

JOBNUM  STATE IPRI JIN  JLIST    INTRODUCED  JOB NAME

#J18    EXEC QUIET  10S LP       FRI  3:18A  ODBCLNSE,MANAGER.SYS
#J250   EXEC        10S LP       SUN  6:37P  STREAMER,MANAGER.SYS
#J248   EXEC        10S LP       SUN  6:36P  JINETD,MANAGER.SYS
#S6881  EXEC       651  651      SAT  1:57A  LEOB,MANAGER.SYS
#J240   EXEC        10S LP       SUN  6:36P  SCOPEJOB,MANAGER.SYS
#S3489  EXEC       769  769      MON 12:46P  RTYLER,MANAGER.SYS

6 JOBS (DISPLAYED):
    0 INTRO
    0 WAIT; INCL 0 DEFERRED
    6 EXEC; INCL 2 SESSIONS
    0 SUSP
JOBFENCE= 7; JLIMIT= 35; SLIMIT= 800
```

I But I wanted only the JOBS, so I pipe the output to the "grep " program and ask it to show me only lines containing the string "SYS":

```
shell/iX> callci "showjob;job=@j" | grep SYS
```

```
#J18    EXEC QUIET  10S LP       FRI  3:18A  ODBCLNSE,MANAGER.SYS
#J250   EXEC        10S LP       SUN  6:37P  STREAMER,MANAGER.SYS
#J248   EXEC        10S LP       SUN  6:36P  JINETD,MANAGER.SYS
#J240   EXEC        10S LP       SUN  6:36P  SCOPEJOB,MANAGER.SYS
```

Now suppose I want to see what output spoolfiles are being used by these jobs? Well that's a little more tricky. We know that SHOWOUT:SP;JOB=<jobnum> will showout these. The jobnum is the first column of the SHOWJOB command. We can extract a columns easily using a program called "awk".

Awk is a complex programming lanquage. The way I use awk most often is to extract a field and add text containing that text. The default field separator is a white space (i.e. space, tab, cr). If we pipe the output of our command to awk, we can extract column 1 using the "$1" variable like this:

```
callci "showjob;job=@j" | grep SYS | awk '{print $1}'
```

```
shell/iX> callci "showjob;job=@j" | grep SYS | awk '{print $1}'
#J18
#J250
#J248
#J240
```

Notice the output is only field 1. We can include other fields and even modify the data using the "printf" function of awk (very similar to the C language printf).

```
callci "showjob;job=@j" | grep SYS | awk '
{printf(" showout;sp;job=%s\n" ,$1)}
'
```

This is interesting, but it only shows us the command. It doesn't actually do it. So we need to send (or pipe?) this command to the CI. Well, piping to CI.PUB.SYS doesn't work. Piping to /SYS.HPBIN/SH almost works…but what works best is the system() function of the awk program:

```
callci "showjob;job=@j" | grep SYS |
awk '{system(" callci \" showout;sp;job=" $1" \" " )}'
```

```
shell/iX> callci "showjob;job=@j" | grep SYS |
awk '{system("callci \"showout;sp;job="$1"\"")}'

DEV/CL   DFID       JOBNUM  FNAME    STATE FRM SPACE RANK PRI #C
LP       #O18       #J18    $STDLIST OPENED       256      1   1
OUTFENCE = 7

DEV/CL   DFID       JOBNUM  FNAME    STATE FRM SPACE RANK PRI #C
LP       #O3971     #J250   $STDLIST OPENED       256      1   1
OUTFENCE = 7

DEV/CL   DFID       JOBNUM  FNAME    STATE FRM SPACE RANK PRI #C
LP       #O3969     #J248   $STDLIST OPENED       512      1   1
OUTFENCE = 7

DEV/CL   DFID       JOBNUM  FNAME    STATE FRM SPACE RANK PRI #C
LP       #O3961     #J240   $STDLIST OPENED       512      1   1
OUTFENCE = 7
shell/iX>
```

Of course, this is way more information than I want; so it's back to grep to filter the data. Grep recognizes regular expressions (which I've so far avoided successfully). Another cool trick of grep is that it has some command switches like : "-i" for ignore case and "-v" for invert argument (i.e NOT this) so I'll use that to remove "DFID and OUTFENCE" so I can just see the data I want:

```
callci "showjob;job=@j" | grep SYS |
```

```
awk ‘{system(“ callci \” showout;sp;job=” $1” \” ” )}’ |
egrep –v “ (DFID|OUTFENCE)”

LP       #O18       #J18    $STDLIST OPENED       256      1   1

LP       #O3971     #J250   $STDLIST OPENED       256      1   1

LP       #O3969     #J248   $STDLIST OPENED       512      1   1

LP       #O3961     #J240   $STDLIST OPENED       512      1   1
```

Ok, this gave me the data I wanted (I cheated and used “egrep” vs “grep” because I needed an expression with “OR” to eliminate the “DFID” OR the “OUTFENCE” lines.

Notice the blank lines? To get rid of them you can add another expression to the list which is “^$” which translated means: “Beginning of line followed immediately by end of line”. The “^” means beginning of line and “$” means end of line. So, all together it looks like this:

```
shell/iX> callci "showjob;job=@j" | grep SYS |
awk '{system("callci \"showout;sp;job="$1"\"")}' |
egrep -v "(DFID|OUTFENCE|^$)"
LP       #O18       #J18    $STDLIST OPENED       256      1   1
LP       #O3971     #J250   $STDLIST OPENED       256      1   1
LP       #O3969     #J248   $STDLIST OPENED       512      1   1
LP       #O3961     #J240   $STDLIST OPENED       512      1   1
shell/iX>
```

This is just a short (rather bizarre) example of some of the power of the POSIX utilities you already have. There are several more that we’ll talk about in the full version of this paper and in the presentation.

Some of the other programs we’ll discuss:

uniq
sort
join
tr
tail
cut
paste
tee
diff
wc

The full version of this paper will be available at http://www.thinkmbs.com