Environment:

The Shell:

* Most programs make few, if any, changes to process attributes (i.e. changeable fields of process data structure) after fork or exec
* One big exception: the shell
* Like all processes, shell is descendant of init
* Shell’s main purpose: run programs first fork-ing then exec-ing named program in child process
* Shell carefully sets up process attributes so that program starts running under precisely defined conditions

Login Procedure:

* Init reads list of terminal lines (hardwired and “pseudo terminal”)
* Init forks child per line
* Child init forks and execs getty for its line
* Grandchild getty offers “login:” prompt
* User types in login ID
* Getty forks and execs login, passing ID as argument
* Great-grandchild login reads user’s line from password file
  + Line includes: encrypted password, home directory, preferred shell
* Login prompts for password and decrypts
* If password is correct, login sets up environment & home directory then execs shell

Execution Environment:

* Interesting aspects of UNIX execution environment:
  + Arguments to new program: argc, argv, envp
  + Environment pointed to by char\*\* environ
  + 3 open “streams”: stdin, stdout, stderr
  + current root & directory
  + 6 UIDs/GIDs (plus supplementary GIDs)
  + Concepts invented mostly for shell job control: process group, controlling terminal, session
* New program given list of strings of the form “variable = value”
* E.g. EDITOR = /usr/pkg/bin/emacs
* All-caps variable name is only a convention
* Environment variable definitions available 3 ways:
  + Global variable char\*\* environ
  + As optional third argument to main: main(int argc, char\* argv[], char\* envp[]) (deprecated)
  + Get env (3)
* Variable length pointer arrays argv, envp/environ are terminated by NULL pointer
* Getenv gets one variable at a time whereas envp and environ point to entire list
* E.g. char\*ed = getenv (EDITOR)
* To see entire environment, that shell will pass to programs: printenv(1)
* You will see some weird stuff… shit gets weird.
* Most definitions made by shell or its ancestors; e.g. login
* Environment variables defined with: $ export EDITOR = /usr/pkg/bin/emacs
* To remove definition $ unset EDITOR
* Process can change own environment but…
* There is no way for child of shell to add to shell’s environment
* Setenv(3) and putenv(3) will add to calling process’ environment
* Unsetenv(3) will remove from calling process’ environment

Shell Export Concept

* Typically think of shell as “program executor” but in fact, it is an interpreter for its on language
* Executing a program is just one of the language’s features
* Another feature: declaration and assignment to variables
* Variable is passed to executed program only if it’s exported
* Non-exported variables still useful for shell programming

Current Root and Directory

* File names beginning with / are interpreted as relative to process’s current root
* Other file names are interpreted relative to process’s current directory
* Only root may chroot(2) (typically after fork and before exec)
* Purpose: limit parts of file system exec-ed program may see; e.g. ftp server