Builtin Commands (1)

- `envir`  
  » Display the environment variables HOME, PWD, PATH

- `cd <pathname>`  
  » Change PWD to `<pathname>`
  » Assume $HOME if `<pathname>` is omitted
  » The user must have permission to cd to `<pathname>` or else “permission denied”

- `echo <word> <word> ...`  
  » Multiple consecutive white spaces can be compressed to a single space

- `quit`

Builtin Commands (2)

- `bg <command>`  
  » Run `<command>` in the background

- `wait`  
  » Wait for all backgrounded processes to complete

- `status`  
  » Display the process id, system time, user time and state (RUNNING or TERMINATED) of the latest processes

- `set <variable> <value>`  
  » The shell replaces all occurrences of `${variable}` with its value
  » Assume `<value>` has no white space (space, tab)
  » Do we need quoting mechanism???

Miscellaneous

- If first word is not a builtin command, it must be a command found in a directory listed in `$PATH`
  » Directory names are separated by colon (":"
Strategy

- What are the requirements?
  - Are there assumptions?
  - Which are the critical requirements?
- What are the risky parts of the project?
- What are the main data structures?
  - i.e., Abstract data types (data and function members)
- What is the overall control flow?
- What system calls and library functions will be needed?
- What should be the development plan?
- Can you sketch the whole project on ONE sheet of paper?
  - Assume simple implementations of each feature

Risk Assessment (1)

- environ
  - From Homework 2 (getenv)
- chdir <pathname>
  - String manipulation of PWD
  - Need easy way of testing for cd permission
- echo <word> <word> ...
  - From Homework 2 (strsep or strtok)
- quit
  - From Homework 2

Risk Assessment (2)

- bg <command>
  - Straightforward fork-execve???
  - What if <command> is a builtin command?
- wait
  - Which processes are in the background?
- status
  - How do we get system time and user time of processes in background? (getrtime)
- set <variable> <value>
  - Symbol table routines will eventually be supplied
  - Symbol table is a binary relation SymTbl: Var → Value
  - Need to concatenate strings during substitution (strncat)

Risk Assessment (3)

- Non-builtin commands
  - Search for command in directories listed in $PATH
  - Command must be accessible; i.e., permissions ok
Observations

- The `chdir(2)` call
  - Every process has a current working directory
  - The search for a relative pathname starts from the current working directory
  - `chdir` can return with an error for many reasons, including incorrect search permission, missing file, etc.
    - See `perror(3)`, `intro(2)`, and `errno(3)`

- The `execve(2)` call
  - Will return with an error for many reasons, including mode not executable, incorrect search permission, etc.

Abstract Data Types

- Symbol table for shell variables
  - Create an empty table of symbol-value pairs
    - `SymTbl : Name → StringValue`
  - Insert a symbol-value pair
  - Get the value of a symbol
    - `[[ Delete a symbol table ]]`

- Command table of arg counts for builtin commands
  - Create a table of command-integer pairs
    - `CmdnTbl : Name → Integer`
  - Statically, initialize with correct pairs
  - Get the number of args for a command
    - Need a convention for commands with variable number of args
    - `[[ Delete a command table ]]`

Basic Control Flow

```c
main:
  Initialize;
  while (get line not EOF) {
    Break line into words;
    if (builtin command) do_builtin;
    else do_nonbuiltin;
  }
do_builtin:
  if (incorrect #args) { Display msg; return; }
  case command {
    ... chdir, environ, echo, quit, bg, wait, status, set ...
  }
do_nonbuiltin:
  if ((pid=Fork()) error) ... Error ...
  if (pid == 0) { // child
    ... execve code ...
    exit(0);
  }
  Wait for child to terminate;
```