```
*-----------------------------------------------------------
* syntax
*  clear editor and extract item b within structure a
*     rextract using filename , item(a(b)) clear
*  extract item b within structure a and store as a matrix
*     rextract using filename , item(a(b)) matrix
*  clear editor and extract item b within structure a,
*  call the new variable k
*     rextract using filename , item(a(b)) clear name(k)
*-----------------------------------------------------------
program rextract , rclass
   syntax using/ , Item(string) [ Matrix name(string) Clear Add ]

* Set the item name(s)
   if `"`name'"' != "" {
      tokenize `"`name'"'
        local vname "`1'"
   }
   else {
      local vname = subinstr("`item'","(","_",.)
      local vname = subinstr("`vname'",".","_",.)
      local vname = subinstr("`vname'",")","",.)
   }
   tokenize `item' , parse("() ")
   local i = 1
   local nlevel = 1
   local level1 "`using'"
   while "``i''" != "" {
      if "``i''" != "(" & "``i''" != ")" {
         local ++nlevel
         local level`nlevel' "``i''"
      }
      local ++i
   }
* find the item & determine its type and size
   local level = 1
   tempname fh
   file open `fh' using "`using'" , read
   file read `fh' inLine
   local inLine  `"`using' = `inLine'"'
   local i = 0
   local depth = 0
   local br  = 0
   local dim = 0
   local found = 0
   local done = 0
   local size ""
   while !r(eof) & `done' == 0 {
      local ++i
      tokenize  `"`inLine'"' , parse(" =()")
      local j = 1
      while  `"``j''"' != "" & `done' == 0 {
         if `dim' == 1 & inlist(`"``j''"',"c","=","(",")",",") == 0 {
            local v = subinstr(`"``j''"',"L","",.)
            local size "`size' `v'"
         }
         if `"``j''"' == ".Dim" local dim = 1
         if  `"``j''"' == "=" & `br' == 0 {
* new item
            local k = `j' - 1
            local h = `j' + 1
            local hh = `j' + 3
            if `found' == 1 local done = 1
            if `done' == 0 {
               if `"``k''"' == "`level`level''" {
                  local ++level
                  if `level' > `nlevel' {
                     local found = 1
                     local size ""
                     local type ""
                  }
               }
               if `"``h''"' == "structure" & `"``hh''"' == "list" {
* item is a structure
                  local type "list"
                  local depth = `depth' + 4
                  local br = -2
               }
               else if `"``h''"' == "quote" local type "string"
               else if `"``h''"' == "c" local type "variable"
               else if `"``h''"' == "structure" & `"``hh''"' == "c" local
type "matrix"
               else local type "scalar"
            }
         }
         else if `"``j''"' == "(" local ++br
         else if `"``j''"' == ")" {
            if `dim' == 1 local dim = 0
            local --br
            if `br' == -2 {
               local depth = `depth' - 4
               local br = 0
            }
         }
         local ++j
      }
      file read `fh' inLine
   }
   file close `fh'
   if `found' == 0 {
      di as error "`item' could not be found in the `using'"
      exit(0)
   }
   else {
      di as txt "`item' found with type `type'" _continue
      if "`size'" != "" di " and size `size'"
      else di ""
        if "`type'" == "matrix" & "`matrix'" == "" di "Data read into
variables"
   }
* re-read the file and find the start of the item
   local level = 1
   tempname fh
   file open `fh' using "`using'" , read
   file read `fh' inLine
   local inLine  `"`using' = `inLine'"'
   local i = 0
   local depth = 0
   local br  = 0
   local found = 0
   local done = 0
   while !r(eof) & `done' == 0 {
      local ++i
      tokenize  `"`inLine'"' , parse(" =(), ")
*di `"(`done':`found')`inLine'"'
      local j = 1
      while  `"``j''"' != "" & `done' == 0 {
         if  `"``j''"' == "=" & `br' == 0 {
* new item
            local k = `j' - 1
            local h = `j' + 1
            local hh = `j' + 3
            if `found' == 1 local done = 1
            if `done' == 0 {
               if `"``k''"' == "`level`level''"  {
                  local ++level
                  if `level' > `nlevel' {
                     local found = 1
* target found: define variables & then read values
                     if "`type'" == "variable" {
                                 local oldN = _N
                        local row = 0
                        if "`clear'" != "" {
                           drop _all
                           qui set obs 1000
                           local grow = "Y"
                        }
                        qui gen `vname' = .

                     }
                     if "`type'" == "matrix" {
                        gettoken r size : size
                        local r = subinstr("`r'",",","",.)
                        gettoken c size : size
                        if "`matrix'" != "" {
                           matrix `vname' = J(`r',`c',0)
                           local row = 0
                           local col = 1
                        }
                        else {
                           local type = "set"
                           if "`clear'" != "" {
                              drop _all
                              qui set obs `r'
                           }
                           forvalues v = 1/`c' {
                              qui gen `vname'_`v' = .
                           }
                           local row = 0
                           local col = 1
                        }
                     }
                  }
               }
               if `"``h''"' == "structure" & `"``hh''"' == "list" {
                  local depth = `depth' + 4
                  local br = -2
               }
            }
         }
         else if `"``j''"' == "(" local ++br
         else if `"``j''"' == ")" {
            if `found' == 1 {
```

```
* finished reading values
               if "`type'" == "variable" & "`clear'" != "" qui drop if _n >
`row'
               file close `fh'
               exit(0)
            }
            local --br
            if `br' == -2 {
               local depth = `depth' - 4
               local br = 0
            }
         }
         else if `found' == 1 & `"``j''"' != ","  & `"``j''"' != "c" &
`"``j''"' != "structure" {
            if "`type'" == "scalar" {
               file close `fh'
               di `"returned as local r(`vname') = ``j''"'
               return local `vname' = `"``j''"'
               exit(0)
            }
            if "`type'" == "variable" {
               local ++row
               if `row' > _N {
                  local length = _N + 1000
                  qui set obs `length'
               }
               qui replace `vname' = ``j'' in `row'
            }
            if "`type'" == "matrix" {
               local ++row
               if `row' > `r' {
                  local row = 1
                  local ++col
               }
               matrix `vname'[`row',`col'] = ``j''
            }
            if "`type'" == "set" {
               local ++row
               if `row' > `r' {
                  local row = 1
                  local ++col
               }
               qui replace `vname'_`col' = ``j'' in `row'
            }
         }
         local ++j
      }
      file read `fh' inLine
   }
   file close `fh'
end

*-----------------------------------------------------------
* syntax
*    rdescribe using filename
*-----------------------------------------------------------
program rdescribe
   syntax using/

   di as txt "Structures and sub-structures within file: " _continue
   tempname fh
   file open `fh' using "`using'" , read
   file read `fh' inLine
   local inLine  `"`using' = `inLine'"'
   local i = 0
   local depth = 0
   local br  = 0
   local dim = 0
   while !r(eof)  {
      local ++i
      tokenize  `"`inLine'"' , parse(" =()")
      local j = 1
      while  `"``j''"' != "" {
         if `dim' == 1 & inlist(`"``j''"',"c","=","(",")",",") == 0 {
            local v = subinstr(`"``j''"',"L","",.)
            di `" `v'"' _continue
         }
         if `"``j''"' == ".Dim" {
            local dim = 1
            di _col(30) " Dim: " _continue
         }
         if  `"``j''"' == "=" & `br' == 0 {
* new item in file
            local k = `j' - 1
            local h = `j' + 1
            local hh = `j' + 3
            di as txt _dup(`depth') " " `" ``k'' "' _continue
            if `"``h''"' == "structure" & `"``hh''"' == "list" {
* item is a structure
               if `depth' != 0 di _column(30) " (list)"
               else di ""
               local depth = `depth' + 4
               local br = -2
            }
            else if `"``h''"' == "quote" di _column(30) " (string)"
            else if `"``h''"' == "c" di _column(30) " (single variable)"
            else if `"``h''"' == "structure" & `"``hh''"' == "c" di
_column(30) " (matrix or set of variables)"
            else di _column(30) " (single value)"
         }
         else if `"``j''"' == "(" local ++br
         else if `"``j''"' == ")" {
            if `dim' == 1 local dim = 0
            local --br
            if `br' == -2 {
               local depth = `depth' - 4
               local br = 0
            }
         }
         local ++j
      }
      file read `fh' inLine
   }
   file close `fh'
end
```