Chapter 13 -- SET AND CRITERIA SELECTION

One of the most powerful characteristics of LTPlus is the capability to define and operate commands upon selected sets of objects. There are 13 **crit_** commands for developing criteria for set selection and 5 **set_** commands for manipulating existing sets. Two additional commands provide the ability to clear all existing **set** criteria and to actually apply existing criteria to the current, active map to produce a working set.

Once a particular set of objects has been created (and assigned a unique set number), that set can be used as an argument with many of the LTPlus commands. One typical use of this capability is in the production of selective plots and reports. Standard sequences of criteria and set selection commands are often saved as macros (*Chapter 16*). **HINT**: Take a good look at the sample set and criteria process at the end of this chapter.

CRITERIA DEVELOPMENT

Criteria elements determine the objects which will be selected when a **crit_exec** command is executed. The criteria elements must be established first, using one or more **crit_** commands. All of the criteria elements so defined must be met before an object is selected and placed in the target **set**.

CRITERIA AND SET COMMAND SEQUENCE

The general procedure for establishing criteria elements and generating a set selection activity is:

crit_clea	Clear all existing criteria
crit_?	Set specific selection criteria
crit_?	Set specific selection criteria
crit_?	Set specific selection criteria
crit_exec	Implement current criteria
set_?	More set manipulation
set_?	More set manipulation
(command) s	Initiate commands using new set s

CLEARING AND EXECUTING CRITERIA

The **crit_** commands (except for **crit_clear** and **crit_exec**) all operate upon on a criteria array file which retains all of the settings between execution of a **crit_clear** and a **crit_exec** command. The **crit_clear** command simply clears all previous criteria settings in that array (a selection based on a cleared criteria array would produce a set containing all of the objects in the current, active map).

The **crit_exec** command initiates a search through the objects of the current, active map and applies all of the current criteria settings against that data to produce a special selection of objects in a named set number.

CRITERIA: POLYGON ADJACENCY

The **crit_adj** command establishes adjacency with the polygon objects in a user-specified set as a set selection criteria. The user supplies the set number whose polygon objects will be evaluated for adjacency. A polygon object will only be selected if it is adjacent to any member of the specified set. Up to 10 **crit_adj** commands may be included in a set selection process; adjacency is established if an evaluated polygon is found to meet this criteria for only one of the multiple **crit_adj** commands included. Assembly is required.

CRITERIA: ATTRIBUTE STRING LEXICAL RANGE

The **crit_arange** command establishes lexical range criteria for evaluating all attributes of all objects. Objects with attributes which fall within the alphabetic range identified by the two user-supplied arguments to this command will be selected. Wildcard characters are permitted. Assembly is required. Up to 10 **crit_arange** elements may be included in the selection; an object will be selected if it is found to meet this criteria for only one of the multiple **crit_arange** commands included. The wildcard characters include:

? = will match any 1 character in attribute
***** = will match any group of characters in attribute
\\= \ in attribute
\?= ? in attribute
*= * in attribute

NOTE: only one * meta-character may be used in the string.

CRITERIA: ATTRIBUTE STRING MATCHING

Yet another command for evaluating object attributes, the **crit_astrg** command sets criteria with regard to matching attribute characters or character sequences with user-specified string data. The input string must match the attribute string of the evaluated object exactly except where special, wildcard characters are included in the user-specified string. To enter a special wildcard character as a legitimate string character, where it is not intended to operate as a wildcard character, it must be preceded by the meta character \. The wildcard characters are the same as those defined above. Assembly is required. Up to 10 **crit_astrg** elements may be included in the selection; an object will be selected if it is found to meet this criteria for only one of the multiple **crit_astrg** commands included.

CRITERIA: ATTRIBUTE STRING LENGTH

The **crit_slen** command establishes length range criteria for evaluating object attribute string lengths. Objects whose attributes have with lengths which fall within the range identified by the two user-supplied arguments to this command will be selected. The length range is specified by a lower and upper bound. Note that by setting both the lower and upper bound as zero (**0**), objects with no attributes will be selected. Assembly is required. Up to 10 **crit_slen** elements may be included in the selection; an object will be selected if it is found to meet this criteria for only one of the multiple **crit_slen** commands included.

CRITERIA: DICTIONARY ATTRIBUTES

The **crit_newatr** command sets criteria for determining whether or not the attributes assigned to all objects on the current, active map have corresponding definitions in the data dictionary identified in the **header** file. Every map created in LTPlus may have an associated dictionary which contains attribute equivalencies. The data dictionary is optional. If one exists, executing a **crit_newatr** command will cause LTPlus to compare all map object attributes with the elements of the data dictionary. Objects with attributes for which no corresponding dictionary entry can be found will be selected. This command is useful for identifying attributing errors. Assembly is required.

CRITERIA: AREA RANGE

The **crit_area** command establishes polygonal area range criteria for evaluating polygon objects. Polygons with areas which fall within the range identified by the two user-supplied arguments to this command will be selected. The area range, specified by a lower and upper bound, is defined further by the area units code also supplied as an argument to the **crit_area** command. Assembly is required. Up to 10 **crit_area** elements may be included in the selection; an object will be selected if it is found to meet this criteria for only one of the multiple **crit_area** commands included. The area units codes include:

0 = pixels
1 = square inches (map)
2 = acres
3 = square miles
4 = square cm (map)
5 = square meters
6 = square kilometers

CRITERIA: LENGTH RANGE

The **crit_length** command establishes length range criteria for evaluating polygon perimeters or line segment lengths. Objects (lines or polygons) with lengths which fall within the range identified by the two user-supplied arguments to this command will be selected. The length range, specified by a lower and upper bound, is defined further by the length units code also supplied as an argument to the **crit_length** command. Assembly is required. **Up to 10 crit_length** elements may be included in the selection; an object will be selected if it is found to meet this criteria for only one of the multiple **crit_length** commands included. The length units codes include:

- **0** = pixels
- **1** = map inches
- **2** = ground feet
- **3** = ground miles
- **4** = map centimeters
- **5** = ground meters
- **6** = ground kilometers

CRITERIA: NUMBER OF SIDES OF A POLYGON
The **crit_psides** command establishes criteria for evaluating the number of sides of polygon objects. Polygons which are composed of a number of sides, the value of which falls within the range identified by the two user-supplied arguments to this command will be selected. The number of sides range is specified by a lower and upper bound. The minimum number of sides (lower bound) must be greater than zero (0) and the maximum number of sides (upper bound) must be greater than or equal to the lower bound value. The range is inclusive. Up to 10 **crit_psides** elements may be included in the selection; an object will be selected if it is found to meet this criteria for only one of the multiple **crit_psides** commands included.

CRITERIA: RECORD NUMBER
The **crit_record** command establishes range criteria for evaluating objects with regard to their record numbers. Objects whose record numbers fall within the range identified by the two user-supplied arguments to this command will be selected. The record number range, specified by a lower and upper bound, is inclusive. Up to 10 **crit_record** elements may be included in the selection; an object will be selected if it is found to meet this criteria for only one of the multiple **crit_record** commands included.

CRITERIA: SERIAL NUMBER
The **crit_serial** command establishes range criteria for evaluating objects with regard to their serial numbers. Objects whose serial numbers fall within the range identified by the two user-supplied arguments to this command will be selected. The serial number range, specified by a lower and upper bound, is inclusive. Up to 10 **crit_serial** elements may be included in the selection; an object will be selected if it is found to meet this criteria for only one of the multiple **crit_serial** commands included.

CRITERIA: OBJECT TYPE
The **crit_type** command establishes type criteria for evaluating objects. Objects which match the category of the type code specified will be selected. Up to 10 **crit_type** elements may be included in the selection; an object will be selected if it is found to meet this criteria for only one of the multiple **crit_type** commands included. The object type codes include:

- **0** = any node
- **1** = any segment
- **2** = any polygon
- **3** = node with 0 branches: an isolated point
- **4** = node with 1 branch: a spur endpoint
- **5** = node with 2 branches: a node on a simple island
- **6** = node with 3 branches: a junction node
- **7** = node with 4 branches: a junction node
- **8** = segment with 2 free (spur) ends: a free segment
- **9** = segment with 1 free (spur) end: a spur

10= segment with no free ends
11= single links to islands (segment)
12= segments between polys with same attribute
13= possible segment contour attr. sequence error
14= interior polygon
15= exterior polygon (negative area)
16= map border exterior polygon (negative area)

CRITERIA: TOPOLOGIC ELEMENT MEMBERSHIP

The **crit_subel** command establishes topologic constraints on objects to be selected. To be selected, an object must be a topologic element of a member of the specified set. A *segment* is an element of a polygon if it makes up part of the boundry of that polygon. A *node* is an element of a segment if it is an endpoint of that segment. A node is an element of a polygon if it is an element of a segment which is an element of that polygon. Every object is a *subelement* of itself. An *exterior* (island perimeter) polygon is a subelement of its parent polygon. Up to 10 **crit_subel** criteria may be specified. If more than one is included in the selection set, only one must be true for an object to be included in the target set. Assembly is required.

The **crit_superel** command establishes topologic constraints on objects to be selected. To be selected, an object must have as an element a member of the specified set. A *segment* is an element of a polygon if it makes up part of the boundry of that polygon. A *node* is an element of a segment if it is an endpoint of that segment. A node is an element of a polygon if it is an element of a segment which is an element of that polygon. Every object is a *superelement* of itself. A normal (interior) polygon (or the map border) is a superelement of its islands. Up to 10 **crit_superel** criteria may be entered. Only one of multiple **crit_superel** entries must be true for an object to be included in the target set.

CRITERIA REPORTING

To generate a report showing all current set selection criteria, the user may execute the **crit_report** command. This command lists all pending selection criteria created by **crit_** commands. The report can be displayed on the screen, printed by the system printer or written to an ascii file. The user must specify the output code as one of the following:

0 = screen
1 = printer
2 = file

If the code included as an argument to this command indicates output to an *ascii* file (2), the user must also supply a file name (with path).

SET MANIPULATION

There are four active **set_** commands which implement, on a named set, the operations *complement*, *exclusion*, *intersection* and *union*. The **set_clear** command clears all named sets for re-use. All set information is carried with a map's data files until cleared with the **set_clear** command.

SET CLEAR

For any given map, the user may define a maximum of 24 sets (assigned numbers 1-24). The **set_clear** command will clear, or erase, the data in the sets defined by the specified range of set numbers. It can also clear import levels in the range 35-95.

SET COMPLEMENT

The **set_comp** command will determine the *complement* of a specified set and store the resulting objects in a second, specified target set. The *complement* of a set is presumed to be all of the current, active map file objects which are not contained in the first set included as an argument to the **set_comp** command. Those objects so identified are stored in the second set specified as an argument to the command. The user may optionally include a target file text descriptor as a third argument to the command.

SET EXCLUSION
The **set_exclude** command compares two user-specified sets and determines which objects are in the first named set but which are not in the second named set. The objects so identified are stored in a third, user-specified set. Any data currently stored in the third, target set will be overwritten. The user may also include a target set text descriptor as an optional fourth argument to the **set_exclude** command.

SET INTERSECTION
The **set_intrsec** command stores all objects which are shared by two, user-specified sets and stores the objects so identified in a third, user-specified target set. The user may also include a target set text descriptor as the fourth, otional argument of the **set_intrsec** command.

SET UNION
Executing the **set_union** command causes the objects in two, user-specified sets to be combined and stored in a third, user-specified target set. The user may also include a text descriptor for the target set as an optional fourth argument to the **set_union** command. Any data previously stored in the target set will be overwritten.

SET REPORTING
To generate a report showing all current sets the names and numbers of elements in all sets, the user may execute the **set_report** command. The report can be displayed on the screen, printed by the system printer or written to an ascii file. The user must specify the output code as one of the following:

0 = screen
1 = printer
2 = file

If the code included as an argument to this command indicates output to an ascii file (2), the user must also supply a file name (with path).

SAMPLE PROCESS
Creating & Using Sets

Set operations are a very important part of the LTPlus environment. As with any powerful tool,this capability is often not used because its operation is not well understood. With the hope of remedying that condition, we offer this discussion regarding the development and use of sets.

First, let's define some terms associated with set operations:

object — this is a unique entity comprised of one or more pixels, points or line segments. The most common objects in LTPlus are points, line segments (contours are in this category) and polygons.

set — a collection of one or more objects. Membership in a particular set is defined by one or more criteria elements.

criteria — a condition which serves to define membership in a set.

range — a condition which is defined as the inclusive span of alphanumeric values which lie on or between a given lower (minimum) bound and an upper (maximum) bound.

wildcard — operations with alphanumeric (string) characters, a wildcard character, like its counterpart in card games, the joker, is an ascii character which implies permission for a specific sort of character substitution, or replacement.

In LTPlus operations, there are two acceptable wildcard characters. The asterisk (*) may represent any size group of ascii characters. The **?** represents a single replacement character. For example, the string **67fg*ab54** represents any string which begins with the characters **67fg** and ends with the character string subset **ab54**. Any number of other ascii characters may be substituted for the *****. Thus, **67fgXab54** and **67fg111222333444ggg666555ab54** can be matched by the wildcard string **67fg*ab54**.

The **?** wildcard character provides matching capability with regard to a single character replacement condition only. The string **67fg?ab54** will only match with another string which begins with **67fg**, ends with **ab54**, and has one ascii character between the two character halves. The string **67fgXab54** matches. The string **67fg111222333ggg666555ab54** does not.

Criteria (**crit_**) commands are used to develop conditions for selecting the membership for a specified (named) set. Once membership has been established, a specified (named) set may be operated in conjunction with another, specified (named) set to produce a third, specified (named) set. And so on, and so on...ad nauseum.

Let's say that you have created, assembled and attributed a map containing several hundred polygons which represent the locations of various species of trees. This map is named **snipe_tree**. You have also digitized a second layer of the same base map area which contains points which represent known red-tailed snipe nesting sites. This map is named **snipe_sites**. Now, you want to develop a report listing the total acreage of areas of ponderosa pine which are contiguous with or contained within the imaginary areas which encircle each snipe location at a radius of 2000 feet each. For safety, you have already made offline copies of both map data sets.

The first step would be to digitally delineate the imaginary snipe circles:

acquire snipe_sites	loads nest site map
crit_clear	clear all criteria

Then make some circular polygons around each nest site position:

mask -1 200	make filled circles
open_solids	opens circles
assemble	lines to polygons

Then, attribute all of these new polygonal areas with the string "**SNIPE**":

set_clear 1 1	clear set 1
crit_type 14	select interior polygons
crit_exec 1 INT_POLYS	store them in set 1
atrbt_set 3 1 SNIPE	attribute w/"**SNIPE**"
save	just for insurance...

Next, merge the two polygon layers into one:

overlay -f snipe_tree	both stored in **snipe_tree**
set_clear 0 1	oh heck...why not?

Identify all of the polygons which have **SNIPE** attributes (some may also have **PP** attribute):

crit_clear	virgo behavior, I admit..
crit_astrg *SNIPE*	find all **SNIPE** polys
crit_exec 1 SNIPE ATTR	put 'em in set 1

Find all of the polygons which are attributed as **Ponderosa Pine** (may also have **SNIPE** as an attribute):

crit_clear	neat..neat..neat..
crit_astrg *PP*	find all **Ponderosa Pines**

crit_exec 2 PINE ATTR	put them in set 2
crit_clear	clean up your act...

Find all occurrences of polygon objects which are adjacent to polygons with the **SNIPE** attribute:

crit_adj 1	adjacent to **SNIPEs**
crit_exec 3 AdjToSNIPE	store this bunch in set 3

Intersect the two sets to form a third set containing only polygons which have both **PP** and SNIPE attributes:

set_intrsec 1 2 4 BOTH	**PP** & **SNIPEs** in set 4

Unite the two sets to form one containing all SNIPE polygons which contain or are adjacent to Ponderosa Pine areas:

set_union 3 4 5 ALL	**PP** & or adj to **SNIPEs**

Print a report showing the total acreage of areas which contain **Ponderosa Pine** AND which are contained within or are adjacent to the **SNIPE** areas:

report 1 2 2 6 5 5	ttl ac **SNIPE&PP** or adj **PP**

Make a real impression by producing a map which shows the **Ponderosa Pine** areas, the **SNIPE** areas and the **PP**-in-or-adjacent-to-**SNIPE** areas:

p_create sn 012003	make sel plt penfile snip
plot -s sn 1 24000 4 250	plot a pretty picture!

That wasn't so hard was it? Naturally, it is important to remember that there are other sequences of operations which can generate the same required information. This may or may not be the most efficient. It does work, however!