**arcto** \( x_1 \ y_1 \ x_2 \ y_2 \ r \) \( \text{arcto} \ xt_1 \ yt_1 \ xt_2 \ yt_2 \)

appends an arc of a circle to the current path, possibly preceded by a straight line segment. Its behavior is identical to that of \( \text{arct} \), except that it also returns the user space coordinates of the two tangent points \((xt_1, yt_1)\) and \((xt_2, yt_2)\) on the operand stack.

\( \text{arcto} \) is not allowed as an element of a user path (see Section 4.6, “User Paths”), whereas \( \text{arct} \) is allowed.

Errors: limitcheck, nocurrentpoint, stackunderflow, typecheck, undefinedresult
See Also: arc, arcn, arct, curveto

**array** \( \text{int array array} \)

creates an array of length \( \text{int} \), each of whose elements is initialized with a null object, and pushes this array on the operand stack. The \( \text{int} \) operand must be a non-negative integer not greater than the maximum allowable array length (see Appendix B). The array is allocated in local or global VM according to the current VM allocation mode (see Section 3.7.2, “Local and Global VM”).

Example

\[
3 \text{ array} \Rightarrow [\text{null null null}]
\]

Errors: limitcheck, rangecheck, stackunderflow, typecheck, VMerror
See Also: [ ], aload, astore, packedarray

**ashow** \( \alpha_x \ \alpha_y \ ) \text{string ashow} –

paints glyphs for the characters of \text{string} in a manner similar to \text{show}; however, while doing so, \text{ashow} adjusts the width of each glyph shown by adding \( \alpha_x \) to the glyph’s \( x \) width and \( \alpha_y \) to its \( y \) width, thus modifying the spacing between glyphs. The numbers \( \alpha_x \) and \( \alpha_y \) are \( x \) and \( y \) displacements in the user coordinate system, not in the glyph coordinate system.

This operator enables fitting a string of text to a specific width by adjusting all the spacing between glyphs by a uniform amount. For a discussion of glyph widths, see Section 5.4, “Glyph Metric Information.”
Example

`/Helvetica findfont 12 scalefont setfont`

14 61 moveto (Normal spacing) show

14 47 moveto 4 0 (Wide spacing) ashow

Errors: invalidaccess, invalidfont, ncurrentpoint, stackunderflow, typecheck
See Also: show, awidthshow, cshow, kshow, widthshow, xshow, xyshow, yshow

**astore** \( \text{any}_0 \ldots \text{any}_{n-1} \) array astore array

stores the objects \( \text{any}_0 \) to \( \text{any}_{n-1} \) from the operand stack into \( \text{array} \), where \( n \) is the length of \( \text{array} \). The astore operator first removes the \( \text{array} \) operand from the stack and determines its length. It then removes that number of objects from the stack, storing the topmost one into element \( n-1 \) of \( \text{array} \) and the bottommost one into element 0. Finally, it pushes \( \text{array} \) back on the stack. Note that an astore operation cannot be performed on packed arrays.

If the value of \( \text{array} \) is in global VM and any of the objects \( \text{any}_0 \) through \( \text{any}_{n-1} \) are composite objects whose values are in local VM, an invalidaccess error occurs (see Section 3.7.2, ”Local and Global VM”).

**Example**

\( (a) \ (bcd) \ (ef) \ 3 \ \text{array astore} \Rightarrow [(a) \ (bcd) \ (ef)] \)

This example creates a three-element array, stores the strings a, bcd, and ef into it as elements 0, 1, and 2, and leaves the array object on the operand stack.

Errors: invalidaccess, stackunderflow, typecheck
See Also: astore, astoreint, astoreint

**atan** num den atan angle

returns the angle (in degrees between 0 and 360) whose tangent is num divided by den. Either num or den may be 0, but not both. The signs of num and den determine the quadrant in which the result will lie: a positive num yields a result in the positive y plane, while a positive den yields a result in the positive x plane. The result is a real number.