==========

SktSocketUser

INHERITS FROM	Object
DECLARED IN	SktSocketUser.h

CLASS DESCRIPTION

The SktSocketUser class, together with the SktSocketManager and SktSocket classes, provides an application with the ability to act as a server for Berkeley UNIX stream socket connections. It can also work with an SktSocket alone in a client application. SktSocketUser is an abstract superclass providing retrieval of an associated SktSocket's input and return of output to that SktSocket for queuing. The actual processing of input must be implemented in a subclass or category.

An SktSocketUser object is created by the application's

SktSocketManager after an SktSocket is created. After that, however, the SktSocketManager has nothing further to do with it. You are responsible for having your subclass process its input, either by sending it update messages in the same style as the SktSocketManager's **update** method, or by using a timed entry within your subclass. Leaving control for an SktSocketUser outside of the SktSocketManager gives the definer of the subclass more control over its actions; you may want to suspend processing on some SktSocketUsers, for example. In each update cycle, the SktSocketUser can simply get an input line (or more), process it, and have its associated SktSocket queue the output.

An SktSocketUser provides several different ways to access its input queue. It can be treated as a simple buffer of characters, from which you always draw a specified amount. It can also be treated as a set of ªlinesº delimited by a character you specify (the default, of course, is linefeed). And last, it can be told on demand to get all input characters up to and including a specific character (byte value). These last two access methods also allow for stripping of the delimiter in the returned data.

For more information on sockets, see: the related UNIX man pages; ªAn Introductory 4.3BSD Interprocess Communication Tutorialº (reprinted in UNIX Programmer's Supplementary Documents Volume 1, PS1:7); or, ªAn Advanced 4.3BSD Interprocess Communication Tutorialº (reprinted in UNIX Programmer's Supplementary Documents Volume 1, PS1:8).

See also: SktSocketManager, SktSocket

INSTANCE VARIABLES

Inherited from Object Class isa;

Declared in SktSocketUser

```
SktSocket  *socket;
BOOL       doesStrip;
BOOL       doesStripCRLF;
char       delimiter;
char       *inputQueue;
long int   queueLength;
long int   queueLimit;
NXZone     *zone;
```

socket	The SktSocketUser object's SktSocket.
doesStrip	Whether to strip the delimiter/requested char on retrieval by line/character.
doesStripCRLF	YES if carriage returns and linefeeds are stripped in addition to any delimiters.
delimiter	The character used to determine what a line is (defaults to linefeed).
inputQueue	Strings waiting to be processed.
queueLength	The length, in bytes, of the input queue.
queueLimit	The maximum number of delimiters allowed in the input queue at any one time.
zone	The zone that the SktSocket was allocated from.

METHOD TYPES

Initializing and freeing a SktSocketUser object

-€initWithSocket:

	-€init -€free
Modifying SktSocketUser attributes	-€setDelimiter: -€delimiter -€setDoesStrip: -€doesStrip -€setDoesStripCRLF: -€doesStripCRLF -€setQueueLimit: -€queueLimit -€setSocket: -€socket
Managing input and output	-€queueInput: -€purgeInput -€queueOutput:ofLength: -€queueOutputString:
Retrieving input	-€getInput:ofLength: -€ungetInput:ofLength: -€getAllInput: -€inputToChar: -€inputToChar:inZone: -€nextInputLine -€nextInputLineinZone:

INSTANCE METHODS

delimiter:

-€(char)**delimiter**

Returns the character used to delimit lines in the **nextInputLine** and **nextInputLineInZone:** methds. The default is the linefeed character (`\n').

See also: -€**delimiter**, -€**nextInputLine,-€nextInputLineInZone:**

doesStrip:

-€(BOOL)**doesStrip**

Returns YES if the SktSocketUser strips the **...ToChar:** character from input requested by **inputToChar:** and **inputToChar:inZone:**, or the delimiter from input requested by **nextInputLine** and **nextInputLineInZone:**. If NO, stripping is not done in those methods. However, if the **...ToChar:** character or the delimiter is either a carriage return or a linefeed, and **doesStripCRLF** is YES, then those will be stripped.

See also: -€**setDoesStrip:**, -€**setDoesStripCRLF: -€doesStripCRLF**

doesStripCRLF

-€(BOOL)**setDoesStripCRLF**

Returns YES if the SktSocketUser strips all trailing carriage returns and linefeeds from input retrieved by any of the **inputToChar...** or **nextInputLine...** methods. This stripping is performed after regular stripping, so that if regular stripping is not done and the character used to retrieve the text is not a carriage return or linefeed, CRLF stripping is blocked from occurring. Returns **self**.

See also: -€**setDoesStripCRLF**, -€**setDoesStrip: -€doesStrip**

free

-€**free**

Deallocates the storage occupied by the SktSocketUser.

getAllInput:

-€(long int)**getInput:**(char **)*input*

Places the entire contents of the input queue in **input*. Returns the number of bytes in the queue. If you want to be sure there is anything in the queue before you try to get anything, send a **queueLength** message.

If a memory reallocation fails, this method returns -1. If the return value is -1, the SktSocketUser has become corrupt, and will crash the process the next time it attempts to alter its input queue. You should either free the SktSocket or terminate the process.

See also: -€**queueLength**, -€**getInput:ofLength:**, -€**ungetInput:ofLength:**, -€**inputToChar:**, -€**inputToChar:inZone:**, -€**nextInputLine**, -€**nextInputLineInZone:**,

getInput:ofLength:

-€(long int)**getInput:**(char *)*input* **ofLength:**(long int)*length*

Removes up to *length* bytes from the input queue and places them in the buffer specified by *input*. Returns the number of bytes actually retrieved (this may be less than the number requested). If you want to be sure there is enough in the queue before you try to get anything, send a **queueLength** message.

If a memory reallocation fails, this method returns -1. If the return value is -1, the SktSocketUser has become corrupt, and will crash the process the next time it attempts to alter its input queue. You should either free the SktSocket or terminate the process.

See also: **-€queueLength**, **-€ungetInput:ofLength:**, **-€getAllInput:**, **-€inputToChar:**, **-€inputToChar:inZone:**, **-€nextInputLine**, **-€nextInputLineInZone:**,

init

-€init

Initializes the socket user to have no SktSocket. Returns **self**.

See also: **-€initWithSocket:**

initWithSocket:

-€initWithocket:(SktSocket *)*aSocket*

Initializes the socket user to use *aSocket* as its SktSocket. The SktSocket also has its user set to the SktSocketUser. This method is the designated initializer for SktSocketUser objects. Returns **self**, or **nil** if the output queue can't be allocated.

See also: **-€init**, **-€setSocket:**, **-€socket**

inputToChar:

-€(char *)**inputToChar:**(char)*aChar*

Removes characters from the input queue up to and including *aChar*, and returns them as a null-terminated string. If regular stripping is

enabled, *aChar* is removed from the string. Further, if CRLF stripping is enabled, any trailing carriage returns or linefeeds are removed. If no string is found, or if the string found is completely stripped, this method returns an empty string.

If a memory reallocation fails, this method returns NULL. If the return value is NULL, the SktSocketUser has become corrupt, and will crash the process the next time it attempts to alter its input queue. You should either free the SktSocket or terminate the process.

See also: -€**getInput:ofLength:**, -€**inputToChar:inZone:**, -€**nextInputLine**, -€**nextInputLineInZone:**, -€**setDoesStrip:**, -€**doesStrip**, -€**setDoesStripCRLF:**, -€**doesStripCRLF**

inputToChar:inZone:

-€(char *)**inputToChar:**(char)*aChar* **inZone:**(NXZone *)*aZone*

Removes characters from the input queue up to and including *aChar*, and returns them as a null-terminated string allocated from *aZone*. If regular stripping is enabled, *aChar* is removed from the string. Further, if CRLF stripping is enabled, any trailing carriage returns or linefeeds are removed. If no string is found, or if the string found is completely stripped, this method returns an empty string.

If a memory reallocation fails, this method returns NULL. If the return value is NULL, the SktSocketUser has become corrupt, and will crash the process the next time it attempts to alter its input queue. You should either free the SktSocket or terminate the process.

See also: -€**getInput:ofLength:**, -€**inputToChar:**, -€**nextInputLine**, -€**nextInputLineInZone:**, -€**setDoesStrip:**, -€**doesStrip**, -€**setDoesStripCRLF:**, -€**doesStripCRLF**

nextInputLine

-€(char *) **nextInputLine**

Removes characters from the input queue up to and including the delimiter, and returns them as a null-terminated string. If regular stripping is enabled, *aChar* is removed from the string. Further, if CRLF stripping is enabled, any trailing carriage returns or linefeeds are removed. If no string is found, or if the string found is completely stripped, this method returns an empty string.

If a memory reallocation fails, this method returns NULL. If the return value is NULL, the SktSocketUser has become corrupt, and will crash the process the next time it attempts to alter its input queue. You should either free the SktSocket or terminate the process.

See also: -€**getInput:ofLength:**, -€**inputToChar:**, -€**inputToChar:inZone:**, -€**nextInputLineInZone:**, -€**setDoesStrip:**, -€**doesStrip**, -€**setDoesStripCRLF:**, -€**doesStripCRLF**

nextInputLineInZone:

-€(char *) **nextInputLineInZone:**(NXZone *)*aZone*

Removes characters from the input queue up to and including the delimiter, and returns them as a null-terminated string allocated from *aZone*. If regular stripping is enabled, *aChar* is removed from the string. Further, if CRLF stripping is enabled, any trailing carriage returns or linefeeds are removed. If no string is found, or if the string found is completely stripped, this method returns an empty string.

If a memory reallocation fails, this method returns NULL. If the return value is NULL, the SktSocketUser has become corrupt, and will crash

the process the next time it attempts to alter its input queue. You should either free the SktSocket or terminate the process.

See also: **-€getInput:ofLength:**, **-€inputToChar:**, **-€inputToChar:inZone:**, **-€nextInputLine**, **-€setDoesStrip:**, **-€doesStrip**, **-€setDoesStripCRLF:**, **-€doesStripCRLF**

purgeInput

-€purgeInput

Empties the input queue. Returns **self**, or **nil** if a memory reallocation fails. If the return value is **nil**, the SktSocketUser has become corrupt, and will crash the process the next time it attempts to alter its input queue. You should either free the SktSocket or terminate the process.

See also: **-€queueInput:ofLength:**

queueInput:ofLength:

-€queueInput:(const char *)*input* **ofLength:**(long int)*length*

Appends *input* to the input queue. If a queue limit is in effect, the number of delimited sequences is counted, and it that number exceeds the queue limit, all subsequent delimited sequences are deleted from the queue, and any fragmentary portion is advanced to immediately follow the delimited sequences. Returns **self**, or **nil** if a memory reallocation fails. If the return value is **nil**, the SktSocketUser has become corrupt, and will crash the process the next time it attempts to alter its input queue. You should either free the SktSocket or terminate the process.

The SktSocketUser's SktSocket object sends this message in its **getInput** method.

See also: -€**setDelimiter:**, -€**delimiter**, -€**nextInputLine**, -€**nextInputLineInZone:**, -€**setQueueLimit:**, -€**queueLimit**

queueLength

-€(long int)**queueLength**

Returns the length, in bytes, of the input queue. Use this method if you want to be sure you'll get enough characters from an invocation of **getInput:ofLength:**.

See also: -€**getInput:ofLength:**

queueLimit

-€(long int)**queueLimit**

Returns the maximum allowable ªlinesº in the queue to *limit*. Whenever new input is queued, the number of delimiters is counted, and if that exceeds the queue limit, all subsequent delimited sequences are deleted from the queue, and any fragmentary portion is advanced to immediately follow the delimited sequences. If the queue limit is 0 or less, no limit checking is performed.

See also: -€**setQueueLimit:**, -€**setDelimiter:**, -€**delimiter**

queueOutput:ofLength:

-€**queueOutput:**(const char *)*output* **ofLength:**(long int)*length*

Forwards *output* to SktSocket's **queueOutput:ofLength:** method. You should always use this method in your application, rather than directly accessing SktSocket's method, since some processing may need to be performed by the SktSocketUser before the SktSocket gets the data. Returns **self**, or **nil** if a memory reallocation fails. If the return value is

nil, the SktSocketUser has become corrupt, and will crash the process the next time it attempts to alter its input queue. You should either free the SktSocket or terminate the process.

See also: **-€queueOutput:ofLength** (SktSocket)

queueOutputString:

-€queueOutputString:(const char *)*aString*

Forwards *output* to SktSocket's **queueOutput:ofLength:** method. You should always use this method in your application, rather than directly accessing SktSocket's method, since some processing may need to be performed by the SktSocketUser before the SktSocket gets the data. Returns **self**, or **nil** if a memory reallocation fails. If the return value is **nil**, the SktSocketUser has become corrupt, and will crash the process the next time it attempts to alter its input queue. You should either free the SktSocket or terminate the process.

See also: **-€queueOutput:ofLength:** (SktSocket)

setDelimiter:

-€setDelimiter:(char)*aChar*

Sets the character used to delimit lines in the **nextInputLine** and **nextInputLineInZone:** methds. Returns **self**.

Changing the delimiter when there is data in the queue and a queue limit is in effect is probably not a good idea.

See also: **-€delimiter**, **-€nextInputLine,-€nextInputLineInZone:**

setDoesStrip:

-€**setDoesStrip:**(BOOL)*flag*

If *flag* is YES, the SktSocketUser will strip the **...ToChar:** character from input requested by **inputToChar:** and **inputToChar:inZone:**, or the delimiter from input requested by **nextInputLine** and **nextInputLineInZone:**. If NO, stripping is not done in those methods. However, if the **...ToChar:** character or the delimiter is either a carriage return or a linefeed, and **doesStripCRLF** is YES, then those will be stripped.

See also: -€**doesStrip**, -€**setDoesStripCRLF:** -€**doesStripCRLF**

setDoesStripCRLF:

-€**setDoesStripCRLF:**(BOOL)*flag*

If *flag* is YES, the SktSocketUser will strip all trailing carriage returns and linefeeds from input retrieved by any of the **inputToChar...** or **nextInputLine...** methods. This stripping is performed after regular stripping, so that if regular stripping is not done and the character used to retrieve the text is not a carriage return or linefeed, CRLF stripping is blocked from occurring. Returns **self**.

See also: -€**doesStripCRLF**, -€**setDoesStrip:** -€**doesStrip**

setQueueLimit:

-€**setQueueLimit:**(long int)*limit*

Sets the maximum allowable ªlinesº in the queue to *limit*. Whenever new input is queued, the number of delimiters is counted, and if that exceeds the queue limit, all subsequent delimited sequences are deleted from the queue, and any fragmentary portion is advanced to

immediately follow the delimited sequences. If the queue limit is set to 0 or less, no limit checking is performed.

Changing the delimiter when there is date in the queue and a queue limit is in effect is probably not a good idea.

See also: **-€queueLimit**, **-€setDelimiter:**, **-€delimiter**

setSocket:

-€(SktSocket *)**setSocket:**(SktSocket *)*aSocket*

Sets the SktSocket to *aSocket*, and returns the previous SktSocket. Also has the SktSocket's user set to the SktSocketUser.

See also: **-€initWithSktSocket:**, **-€socket:**, **-€setUser:** (SktSocket)

socket

-€(SktSocket *)**socket**

Returns the SktSocketUser's SktSocket's object.

See also: **-€initWithSktSocket:**, **-€setSocket:**

ungetInput:ofLength:

-€(long int)**ungetInput:**(char *)*input* **ofLength:**(long int)*length*

Adds *length* bytes from *input* to the front of the input queue. Use this method to undo the effects of **getInput:ofLength:**. Do not use this with any other retrieval methodÐthey may strip characters from their returned input, which will alter the content of the queue if that data is pushed back. Returns **self**, or **nil** if a memory reallocation fails. If the return value is **nil**, the SktSocketUser has become corrupt, and will crash the process the next time it attempts to alter its input queue. You

should either free the SktSocket or terminate the process.

See also: **-€getInput:ofLength:**