```
**********************************************************
    9714 Thu Jun  5 16:19:50 2014
new/usr/src/man/man9f/string.9f
4849 strlcpy(9F) points to the wrong header
**********************************************************
'\" te
.\" Copyright (c) 2009, Sun Microsystems, Inc.  All Rights Reserved.
.\" The contents of this file are subject to the terms of the Common Development
.\" You can obtain a copy of the license at usr/src/OPENSOLARIS.LICENSE or http:
.\" When distributing Covered Code, include this CDDL HEADER in each file and in
.TH STRING 9F "Jun 4, 2014"
.TH STRING 9F "Feb 27, 2009"
.SH NAME
string, strcasecmp, strncasecmp, strncat, strlcat, strchr, strrchr, strcmp,
strncmp, strcpy, strncpy, strlcpy, strfree, strspn, strdup, ddi_strdup, strlen,
strnlen \- string operations
.SH SYNOPSIS
.LP
.nf
#include <sys/ddi.h>
#include <sys/sunddi.h>
#endif /* ! codereview */

\fBint\fR \fBstrcasecmp\fR(\fBconst char *\fR\fIs1\fR, \fBconst char *\fR\fIs2\f
.fi

.LP
.nf
\fBint\fR \fBstrncasecmp\fR(\fBconst char *\fR\fIs1\fR, \fBconst char *\fR\fIs2\
.fi

.LP
.nf
\fBchar *\fR\fBstrncat\fR(\fBchar *\fR \fIs1\fR, \fBconst char *\fR \fIs2\fR, \f
.fi

.LP
.nf
\fBsize_t\fR \fBstrlcat\fR(\fBchar *\fR\fIdst\fR, \fBconst char *\fR\fIsrc\fR, \
.fi

.LP
.nf
\fBchar *\fR\fBstrchr\fR(\fBconst char *\fR\fIstr\fR, \fBint\fR \fIchr\fR);
.fi

.LP
.nf
\fBchar *\fR\fBstrrchr\fR(\fBconst char *\fR\fIstr\fR, \fBint\fR \fIchr\fR);
.fi

.LP
.nf
\fBint\fR \fBstrcmp\fR(\fBconst char *\fR\fIs1\fR, \fBconst char *\fR\fIs2\fR);
.fi

.LP
.nf
\fBint\fR \fBstrncmp\fR(\fBconst char *\fR\fIs1\fR, \fBconst char *\fR\fIs2\fR,
.fi

.LP
.nf
\fBchar *\fR\fBstrcpy\fR(\fBchar *\fR \fIdst\fR, \fBconst char *\fR \fIsrc\fR);
.fi
```



```
.LP
.nf
\fBchar *\fR\fBstrncpy\fR(\fBchar *\fR \fIdst\fR, \fBconst char *\fR \fIsrc\fR,
.fi

.LP
.nf
\fBsize_t\fR \fBstrlcpy\fR(\fBchar *\fR\fIdst\fR, \fBconst char *\fR\fIsrc\fR, \
.fi

.LP
.nf
\fBvoid\fR \fBstrfree\fR(\fBchar *\fR\fIs\fR);
.fi

.LP
.nf
\fBsize_t\fR \fBstrspn\fR(\fBconst char *\fR\fIs1\fR, \fBconst char *\fR\fIs2\fR
.fi

.LP
.nf
\fBchar *\fR\fBstrdup\fR(\fBconst char *\fR\fIs1\fR);
.fi

.LP
.nf
\fBchar *\fR\fBddi_strdup\fR(\fBconst char *\fR\fIs1\fR, \fBint\fR \fIflag\fR);
.fi

.LP
.nf
\fBsize_t\fR \fBstrlen\fR(\fBconst char *\fR\fIs\fR);
.fi

.LP
.nf
\fBsize_t\fR \fBstrnlen\fR(\fBconst char *\fR\fIs\fR, \fBsize_t\fR \fIn\fR);
.fi

.SH INTERFACE LEVEL
.sp
.LP
Solaris DDI specific (Solaris DDI).
.SH DESCRIPTION
.sp
.LP
The arguments \fIs\fR, \fIs1\fR, and \fIs2\fR point to strings (arrays of
characters terminated by a null character). The \fBstrcat()\fR,
\fBstrncat()\fR, \fBstrlcat()\fR, \fBstrcpy()\fR, \fBstrncpy()\fR,
\fBstrlcpy()\fR, and \fBstrfree()\fR functions all alter their first argument.
Additionally, the \fBstrcpy()\fR function does not check for overflow of the
array.
.SS "\fBstrcasecmp()\fR, \fBstrncasecmp()\fR"
.sp
.LP
The \fBstrcasecmp()\fR and \fBstrncasecmp()\fR functions are case-insensitive
versions of  \fBstrcmp()\fR and \fBstrncmp()\fR respectively, described below.
They assume the \fBASCII\fR character set and ignore differences in case when
comparing lower and upper case characters.
.SS "\fBstrncat()\fR, \fBstrlcat()\fR"
.sp
.LP
The \fBstrncat()\fR function appends at most \fIn\fR characters of string
\fIs2\fR, including the terminating null character, to the end of string
\fIs1\fR. It returns a pointer to the null-terminated result. The initial
```

```
character of  \fIs2\fR overrides the null character at the end of \fIs1\fR. If
copying takes place between objects that overlap, the behavior of
\fBstrncat()\fRand \fBstrlcat()\fR is undefined.
.sp
.LP
The \fBstrlcat()\fR function appends  at most
(\fIdstsize\fR-\fBstrlen\fR(\fIdst\fR)-1) characters of \fIsrc\fR to \fIdst\fR
(\fIdstsize\fR being the  size of the  string buffer \fIdst\fR). If the string
pointed to by \fIdst\fR contains a null-terminated string that fits into
\fIdstsize\fR bytes when \fBstrlcat()\fR is called, the string pointed to by
\fIdst\fR will be a null-terminated string that fits in \fIdstsize\fR bytes
(including the terminating null character) when it completes, and the initial
character of \fIsrc\fR will override the null character at  the end of
\fIdst\fR. If the string pointed to by \fIdst\fR is longer than \fIdstsize\fR
bytes when \fBstrlcat()\fR is called, the string pointed to by \fIdst\fR will
not be changed. The function returns
\fBmin\fR{\fIdstsize\fR,\fBstrlen\fR(\fIdst\fR)}+\fBstrlen\fR(\fIsrc\fR).
Buffer overflow can be checked as  follows:
.sp
.in +2
.nf
if (strlcat(dst, src, dstsize) >= dstsize)
        return \(mi1;
.fi
.in -2

.SS "\fBstrchr()\fR, \fBstrrchr()\fR"
.sp
.LP
The \fBstrchr()\fR function returns a pointer to the first occurrence of
\fIc\fR (converted to a  \fBchar\fR) in string \fIs\fR, or a null pointer if
\fIc\fR does not occur in the string. The \fBstrrchr()\fR function returns a
pointer to the last occurrence of \fIc\fR. The null character terminating a
string is considered to be part of the string.
.SS "\fBstrcmp()\fR, \fBstrncmp()\fR"
.sp
.LP
The \fBstrcmp()\fR function compares two strings byte-by-byte, according to the
ordering of your machine's character set.  The function returns an integer
greater than, equal to, or less than 0, if  the string pointed to by \fIs1\fR
is greater than, equal to, or less than the string pointed to by \fIs2\fR
respectively. The sign of a non-zero return value is determined  by the sign of
the difference between the values of the first pair of bytes that differ in the
strings being compared. The \fBstrncmp()\fR function makes the same comparison
but looks at a maximum of \fIn\fR bytes. Bytes following a null byte are not
compared.
.SS "\fBstrcpy()\fR, \fBstrncpy()\fR, \fBstrlcpy()\fR"
.sp
.LP
The \fBstrcpy()\fR function copies string \fIs2\fR to \fIs1\fR, including the
terminating null character, stopping after the null character has been copied.
The \fBstrncpy()\fR function copies exactly \fIn\fR bytes, truncating \fIs2\fR
or adding null characters to \fIs1\fR if necessary. The result will not be
null-terminated if the length of \fIs2\fR is \fIn\fR or more. Each function
returns \fIs1\fR.  If copying takes place between objects that overlap, the
behavior of \fBstrcpy()\fR, \fBstrncpy()\fR, and \fBstrlcpy()\fR is undefined.
.sp
.LP
The \fBstrlcpy()\fR function copies  at most \fIdstsize\fR\(mi1 characters
(\fIdstsize\fR being the  size of the  string buffer \fIdst\fR) from \fIsrc\fR
to \fIdst\fR,  truncating \fIsrc\fR if necessary.  The  result is always
null-terminated. The function returns \fBstrlen\fR(\fIsrc\fR). Buffer overflow
can be checked as  follows:
.sp
.in +2
.nf
```



```
if (strlcpy(dst, src, dstsize) >= dstsize)
        return \(mi1;
.fi
.in -2

.SS "\fBstrfree()\fR"
.sp
.LP
The \fBstrfree()\fR function frees the memory associated with the string
pointed to by \fIs\fR.  This memory pointed to by \fIs\fR must be of size
\fBstrlen\fR(\fIs\fR)+1, and must have been allocated (either directly or
indirectly) by \fBkmem_alloc\fR(9F) or \fBkmem_zalloc\fR(9F).
.SS "\fBstrspn()\fR"
.sp
.LP
The \fBstrspn()\fR function returns the length of the initial segment of string
\fIs1\fR that consists entirely of characters from string \fIs2\fR.
.SS "\fBstrdup()\fR, \fBddi_strdup()\fR"
.sp
.LP
The \fBddi_strdup()\fR function returns a pointer to a new string that is a
duplicate of the string pointed to by \fIs1\fR. The returned pointer can be
passed to \fBstrfree()\fR or \fBkmem_free\fR(9F). The space for the new string
is obtained using \fBkmem_alloc()\fR. flag can be either \fBKM_SLEEP\fR or
\fBKM_NOSLEEP\fR, and determines whether the caller can sleep for memory.
\fBKM_SLEEP\fR allocations may sleep but are guaranteed to succeed.
\fBKM_NOSLEEP\fR allocations are guaranteed not to sleep but may fail (return
\fINULL\fR) if no memory is currently available.
.sp
.LP
The \fBstrdup()\fR function behaves the same as the \fBddi_strdup()\fR when
called with the \fBKM_SLEEP\fR flag. This means that \fBstrdup()\fR can sleep
until memory is available and will always succeed.
.SS "\fBstrlen()\fR, \fBstrnlen()\fR"
.sp
.LP
The \fBstrlen()\fR function returns the number of bytes in \fIs\fR, not
including the terminating null character.
.sp
.LP
The \fBstrnlen()\fR function returns the smaller of \fIn\fR or the number of
bytes in \fIs\fR, not including the terminating null character. The
\fBstrnlen()\fR function never examines more than \fIn\fR bytes of the string
pointed to by \fIs\fR.
.SH CONTEXT
.sp
.LP
The \fBstrdup()\fR and \fBddi_strdup()\fR functions can be called from user or
kernel context.
.sp
.LP
The \fBddi_strdup()\fR function can be called from interrupt context only if
the \fBKM_NOSLEEP\fR flag is set.
.sp
.LP
All the other string manipulation functions can be called from user, interrupt,
or kernel context.
.SH ATTRIBUTES
.sp
.LP
See \fBattributes\fR(5) for descriptions of the following attributes:
.sp

.sp
.TS
box;
```

```
c | c
l | l .
ATTRIBUTE TYPE  ATTRIBUTE VALUE
_
Interface Stability     Committed
.TE

.SH SEE ALSO
.sp
.LP
\fBstring\fR(3C), \fBattributes\fR(5), \fBbcopy\fR(9F), \fBddi_copyin\fR(9F),
\fBkmem_alloc\fR(9F)
.sp
.LP
\fIWriting Device Drivers\fR
.SH NOTES
.sp
.LP
If copying takes place between objects that overlap, the behavior of
\fBstrlcat()\fR, \fBstrncat()\fR, \fBstrcpy()\fR, \fBstrlcpy()\fR, and
\fBstrncpy()\fR is undefined.
```