

```

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

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

21 .LP
22 .nf
23 \fBint\fR \fBstrncasecmp\fR(\fBconst char *\fR\fIs1\fR, \fBconst char *\fR\fIs2\
24 .fi

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

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

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

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

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

51 .LP
52 .nf
53 \fBint\fR \fBstrncmp\fR(\fBconst char *\fR \fIs1\fR, \fBconst char *\fR \fIs2\fR,
54 .fi

56 .LP
57 .nf
58 \fBchar *\fR\fBstrcpy\fR(\fBchar *\fR \fIdst\fR, \fBconst char *\fR \fIsrc\fR);
59 .fi

```

```

61 .LP
62 .nf
63 \fBchar *\fR\fBstrncpy\fR(\fBchar *\fR \fIdst\fR, \fBconst char *\fR \fIsrc\fR,
64 .fi

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

71 .LP
72 .nf
73 \fBvoid\fR \fBstrfree\fR(\fBchar *\fR \fIs\fR);
74 .fi

76 .LP
77 .nf
78 \fBsize_t\fR \fBstrspn\fR(\fBconst char *\fR \fIs1\fR, \fBconst char *\fR \fIs2\fR
79 .fi

81 .LP
82 .nf
83 \fBchar *\fR\fBstrdup\fR(\fBconst char *\fR \fIs1\fR);
84 .fi

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

91 .LP
92 .nf
93 \fBsize_t\fR \fBstrlen\fR(\fBconst char *\fR \fIs\fR);
94 .fi

96 .LP
97 .nf
98 \fBsize_t\fR \fBstrlen\fR(\fBconst char *\fR \fIs\fR, \fBsize_t\fR \fIn\fR);
99 .fi

101 .SH INTERFACE LEVEL
102 .sp
103 .LP
104 Solaris DDI specific (Solaris DDI).
105 .SH DESCRIPTION
106 .sp
107 .LP
108 The arguments \fIs\fR, \fIs1\fR, and \fIs2\fR point to strings (arrays of
109 characters terminated by a null character). The \fBstrcat()\fR,
110 \fBstrncat()\fR, \fBstrlcat()\fR, \fBstrcpy()\fR, \fBstrncpy()\fR,
111 \fBstrlcpy()\fR, and \fBstrfree()\fR functions all alter their first argument.
112 Additionally, the \fBstrcpy()\fR function does not check for overflow of the
113 array.
114 .SS "\fBstrcasecmp()\fR, \fBstrncasecmp()\fR"
115 .sp
116 .LP
117 The \fBstrcasecmp()\fR and \fBstrncasecmp()\fR functions are case-insensitive
118 versions of \fBstrcmp()\fR and \fBstrncmp()\fR respectively, described below.
119 They assume the \fBASCII\fR character set and ignore differences in case when
120 comparing lower and upper case characters.
121 .SS "\fBstrncat()\fR, \fBstrlcat()\fR"
122 .sp
123 .LP
124 The \fBstrncat()\fR function appends at most \fIn\fR characters of string
125 \fIs2\fR, including the terminating null character, to the end of string
126 \fIs1\fR. It returns a pointer to the null-terminated result. The initial

```

```

127 character of \fIs2\fR overrides the null character at the end of \fIs1\fR. If
128 copying takes place between objects that overlap, the behavior of
129 \fBstrncat()\fR and \fBstrlcat()\fR is undefined.
130 .sp
131 .LP
132 The \fBstrlcat()\fR function appends at most
133 (\fIdstsize\fR-\fBstrlen\fR(\fIdst\fR)-1) characters of \fIsrc\fR to \fIdst\fR
134 (\fIdstsize\fR being the size of the string buffer \fIdst\fR). If the string
135 pointed to by \fIdst\fR contains a null-terminated string that fits into
136 \fIdstsize\fR bytes when \fBstrlcat()\fR is called, the string pointed to by
137 \fIdst\fR will be a null-terminated string that fits in \fIdstsize\fR bytes
138 (including the terminating null character) when it completes, and the initial
139 character of \fIsrc\fR will override the null character at the end of
140 \fIdst\fR. If the string pointed to by \fIdst\fR is longer than \fIdstsize\fR
141 bytes when \fBstrlcat()\fR is called, the string pointed to by \fIdst\fR will
142 not be changed. The function returns
143 \fBmin\fR{\fIdstsize\fR,\fBstrlen\fR(\fIdst\fR)}+\fBstrlen\fR(\fIsrc\fR).
144 Buffer overflow can be checked as follows:
145 .sp
146 .in +2
147 .nf
148 if (strlcat(dst, src, dstsize) >= dstsize)
149     return \mil;
150 .fi
151 .in -2

153 .SS "\fBstrchr()\fR, \fBstrrchr()\fR"
154 .sp
155 .LP
156 The \fBstrchr()\fR function returns a pointer to the first occurrence of
157 \fIc\fR (converted to a \fBchar\fR) in string \fIs\fR, or a null pointer if
158 \fIc\fR does not occur in the string. The \fBstrrchr()\fR function returns a
159 pointer to the last occurrence of \fIc\fR. The null character terminating a
160 string is considered to be part of the string.
161 .SS "\fBstrcmp()\fR, \fBstrncmp()\fR"
162 .sp
163 .LP
164 The \fBstrcmp()\fR function compares two strings byte-by-byte, according to the
165 ordering of your machine's character set. The function returns an integer
166 greater than, equal to, or less than 0, if the string pointed to by \fIs1\fR
167 is greater than, equal to, or less than the string pointed to by \fIs2\fR
168 respectively. The sign of a non-zero return value is determined by the sign of
169 the difference between the values of the first pair of bytes that differ in the
170 strings being compared. The \fBstrncmp()\fR function makes the same comparison
171 but looks at a maximum of \fIn\fR bytes. Bytes following a null byte are not
172 compared.
173 .SS "\fBstrcpy()\fR, \fBstrncpy()\fR, \fBstrncpy()\fR, \fBstrlcpy()\fR"
174 .sp
175 .LP
176 The \fBstrcpy()\fR function copies string \fIs2\fR to \fIs1\fR, including the
177 terminating null character, stopping after the null character has been copied.
178 The \fBstrncpy()\fR function copies exactly \fIn\fR bytes, truncating \fIs2\fR
179 or adding null characters to \fIs1\fR if necessary. The result will not be
180 null-terminated if the length of \fIs2\fR is \fIn\fR or more. Each function
181 returns \fIs1\fR. If copying takes place between objects that overlap, the
182 behavior of \fBstrcpy()\fR, \fBstrncpy()\fR, and \fBstrlcpy()\fR is undefined.
183 .sp
184 .LP
185 The \fBstrlcpy()\fR function copies at most \fIdstsize\fR(\fMil characters
186 (\fIdstsize\fR being the size of the string buffer \fIdst\fR) from \fIsrc\fR
187 to \fIdst\fR, truncating \fIsrc\fR if necessary. The result is always
188 null-terminated. The function returns \fBstrlen\fR(\fIsrc\fR). Buffer overflow
189 can be checked as follows:
190 .sp
191 .in +2
192 .nf

```

```

193 if (strlcpy(dst, src, dstsize) >= dstsize)
194     return \mil;
195 .fi
196 .in -2

198 .SS "\fBstrfree()\fR"
199 .sp
200 .LP
201 The \fBstrfree()\fR function frees the memory associated with the string
202 pointed to by \fIs\fR. This memory pointed to by \fIs\fR must be of size
203 \fBstrlen\fR(\fIs\fR)+1, and must have been allocated (either directly or
204 indirectly) by \fBkmem_alloc\fR(9F) or \fBkmem_zalloc\fR(9F).
205 .SS "\fBstrspn()\fR"
206 .sp
207 .LP
208 The \fBstrspn()\fR function returns the length of the initial segment of string
209 \fIs1\fR that consists entirely of characters from string \fIs2\fR.
210 .SS "\fBstrdup()\fR, \fBbdi_strdup()\fR"
211 .sp
212 .LP
213 The \fBbdi_strdup()\fR function returns a pointer to a new string that is a
214 duplicate of the string pointed to by \fIs1\fR. The returned pointer can be
215 passed to \fBstrfree()\fR or \fBkmem_free\fR(9F). The space for the new string
216 is obtained using \fBkmem_alloc()\fR. flag can be either \fBKM_SLEEP\fR or
217 \fBKM_NOSLEEP\fR, and determines whether the caller can sleep for memory.
218 \fBKM_SLEEP\fR allocations may sleep but are guaranteed to succeed.
219 \fBKM_NOSLEEP\fR allocations are guaranteed not to sleep but may fail (return
220 \fINULL\fR) if no memory is currently available.
221 .sp
222 .LP
223 The \fBstrdup()\fR function behaves the same as the \fBbdi_strdup()\fR when
224 called with the \fBKM_SLEEP\fR flag. This means that \fBstrdup()\fR can sleep
225 until memory is available and will always succeed.
226 .SS "\fBstrlen()\fR, \fBstrnlen()\fR"
227 .sp
228 .LP
229 The \fBstrlen()\fR function returns the number of bytes in \fIs\fR, not
230 including the terminating null character.
231 .sp
232 .LP
233 The \fBstrnlen()\fR function returns the smaller of \fIn\fR or the number of
234 bytes in \fIs\fR, not including the terminating null character. The
235 \fBstrnlen()\fR function never examines more than \fIn\fR bytes of the string
236 pointed to by \fIs\fR.
237 .SH CONTEXT
238 .sp
239 .LP
240 The \fBstrdup()\fR and \fBbdi_strdup()\fR functions can be called from user or
241 kernel context.
242 .sp
243 .LP
244 The \fBbdi_strdup()\fR function can be called from interrupt context only if
245 the \fBKM_NOSLEEP\fR flag is set.
246 .sp
247 .LP
248 All the other string manipulation functions can be called from user, interrupt,
249 or kernel context.
250 .SH ATTRIBUTES
251 .sp
252 .LP
253 See \fBAttributes\fR(5) for descriptions of the following attributes:
254 .sp
255 .sp
256 .sp
257 .TS
258 box;

```

```
259 c | c
260 l | l .
261 ATTRIBUTE TYPE ATTRIBUTE VALUE
262 -
263 Interface Stability Committed
264 .TE

266 .SH SEE ALSO
267 .sp
268 .LP
269 \fBstring\fR(3C), \fBattributes\fR(5), \fBbcopy\fR(9F), \fBbdi_copyin\fR(9F),
270 \fBkmem_alloc\fR(9F)
271 .sp
272 .LP
273 \fIWriting Device Drivers\fR
274 .SH NOTES
275 .sp
276 .LP
277 If copying takes place between objects that overlap, the behavior of
278 \fBstrlcat()\fR, \fBstrncat()\fR, \fBstrcpy()\fR, \fBstrncpy()\fR, and
279 \fBstrncpy()\fR is undefined.
```