sstrprefix 2019-10-01 10:52:54

```
/***   s_str_prefix.c
       2005.9.25
****/

#include "lisp.h"

extern ptr k_end,k_ci,k_nocase;
extern unsigned char *jindex();

static int nocase;

/****   jindex-> byte index  ***/
int jindex_bindex(unsigned char *p,int ind, int le
        n)
{
  int bind=0;
  unsigned char b;
  for(;;){
    /*** printf("ind %d len %d\n",ind,len); ***/
    if(len==0){
          if(ind==0) return bind;
          return -1;  /*** error!  ***/
     }
    if(len<0) {
          return -1;  /*** Err **/
    }
    if(ind==0) return bind;
    b=*p++;
    len--;
    bind++;
    if(euc_first(b)&&euc_code(b,*p)){
           p++;
           len--;
           bind++;
     }
    ind--;
  }
}

/****   bindex-> jindex  ***/
int bindex_jindex(unsigned char *p,int ind, int le
        n)
{
  int jind=0;
  unsigned char b;
  for(;;){
    /*** printf("ind %d len %d\n",ind,len); ***/
    if(len==0){
          if(ind==0) return jind;
          return -1;  /*** error! however Not occu
          r in kmode ***/
     }
    if(len<0) {
          return jind-1;  /*** Err ?? **/
    }
    if(ind==0) return jind;
    if(ind<0) return jind-1; /*** Caution ***/
    b=*p++;
    len--;
    ind--;
    if(euc_first(b)&&euc_code(b,*p)){
           p++;
           len--;
           ind--;
     }
    jind++;
  }
}



/*** return length eq **/
int prefix_cmp_len(Reg unsigned char *p,Reg unsign
        ed char *q,int n,int m){
 Reg int k=n,len=0;
  if(m<n) k=m;
  len=k;
   while(k--){
       if(*p++!=*q++)  return len-k-1;
   }
    return len;
}

int prefix_cmp_len_ci(Reg unsigned char *p,Reg uns
        igned char *q,int n,int m){
 Reg int k=n,len=0;
  if(m<n) k=m;
  len=k;
   while(k--){
       if(tolower(*p++)!=tolower(*q++)) return len
       -k-1;
```

```
   }
    return len;
}

int suffix_cmp_len(Reg unsigned char *p,Reg unsign
        ed  char *q,int n,int m){
 Reg int k=n,len=0;
  if(m<n) k=m;
  p+=n;
  q+=m;
  len=k;
   while(k--){
       if(*--p!=*--q)  return len-k-1;
   }
    return len;
}

int suffix_cmp_len_ci(Reg unsigned char *p,Reg uns
        igned char *q,int n,int m){
 Reg int k=n,len=0;
  if(m<n) k=m;
  p+=n;
  q+=m;
  len=k;
   while(k--){
       if(tolower(*--p)!=tolower(*--q)) return len
       -k-1;
   }
    return len;
}

static key_ci(int narg, ptr *p){
  ptr x;
  while(narg>0){
   x=*p++;
   if(x==k_ci) nocase=1;
   else if(x==k_nocase) nocase=1;
   else
     error(E_range,x);
   narg--;
   }
}




void get_s1_s2(int narg,int len1,int len2,int *s1,
        int *e1,int *s2,int *e2)
{
  int index,k,end,kmode=kanji_mode,n;
  ptr *base,start,endp;
  nocase=0;
     *s1=0;
     *s2=0;
     *e1=len1;
     *e2=len2;
  if(narg==0)  return;
  if(narg>8)
     checknarg(narg-8);
  base=stack-narg;
  start=*base++;
  narg--;
  end=len1;
  if(SINTP(start))
     *s1=SINT_NUMB(start);
  else if(start==k_end){
     if((narg>0)&&(SINTP(*base))) {
         n=SINT_NUMB(*base);
         if(n<0){
            end+=n;
            base++;
            narg--;
         }
     }
     *s1=end;
 }
 else{
       key_ci(narg+1,base-1);
        return;
  }
  if(narg==0){
      *s2=0;
     *e1=len1;
     *e2=len2;
      return;
   }
  else if(narg<0) checknarg(narg);
  endp=*base++;
  narg--;
  end=len1;
  if(SINTP(endp))
        *e1=SINT_NUMB(endp);
  else if(endp==k_end){
      if((narg>0)&&(SINTP(*base))) {
         n=SINT_NUMB(*base);
```

```
        if(n<0){
           end+=n;
           base++;
           narg--;
        }
    }
    *e1=end;
 }
 else{
       key_ci(narg+1,base-1);
        return;
}
  if(narg==0){
      *s2=0;
    *e2=len2;
     return;
   }
/****  string2 ***/
 start=*base++;
  narg--;
  end=len2;
  if(SINTP(start))
    *s2=SINT_NUMB(start);
  else if(start==k_end){
    if((narg>0)&&(SINTP(*base))) {
         n=SINT_NUMB(*base);
        if(n<0){
           end+=n;
           base++;
           narg--;
        }
    }
    *s2=end;
 }
 else{
      key_ci(narg+1,base-1);
       return;
}
  if(narg==0){
    *e2=len2;
     return;
   }
  else if(narg<0) checknarg(narg);
  endp=*base++;
  narg--;
  end=len2;
  if(SINTP(endp))
       *e2=SINT_NUMB(endp);
  else if(endp==k_end){
     if((narg>0)&&(SINTP(*base))) {
        n=SINT_NUMB(*base);
        if(n<0){
           end+=n;
           base++;
           narg--;
        }
    }
    *e2=end;
 }
 else
      ;
   key_ci(narg+1,base-1);
  return ;
}

void set_string2(int narg,unsigned char **pp,unsig
        ned char **qq, int *size1,int *size2,int *
        kanji){
  int len1,len2,end1,end2,kmode=kanji_mode;
  int s1,s2;
  ptr str1,str2,*base;
  unsigned char *p,*q,*r;
  base=stack-narg;
  if(narg<2)
   checknarg(narg-2);
  str1=*base++;
  if (SREGP(str1)||!STRINGP(str1)) error(E_notstr,
  str1);
   p=get_string(str1,&len1);
  str2=*base++;
  if (SREGP(str2)||!STRINGP(str2)) error(E_notstr,
  str2);
   q=get_string(str2,&len2);
   narg-=2;
    end1=len1;
    end2=len2;
    if(kmode){
        kmode=2;
        end1=jstrlength(p,len1);
          if((end1==-1)||(end1==len1)) {
               end1=len1;
               kmode--;
          }
        end2=jstrlength(q,len2);
```

```
          if((end2==-1)||(end2==len2)) {
               end2=len2;
               kmode--;
          }
        }
   get_s1_s2( narg,end1,end2,&s1,&end1,&s2,&end2);
 /*** printf("end1 %d end2 %d len1 %d len2 %d\n",e
        nd1,end2,len1,len2);  ***/
 if(kmode){
      p=jindex(p,s1,len1);
      r=jindex(p,end1-s1,len1);
     len1=r-p;
      q=jindex(q,s2,len2);
     len2=jindex(q,end2-s2,len2)-q;
   }
   else{
     p+=s1;
    len1=end1-s1;
    q+=s2;
    len2=end2-s2;
   }
  *pp=p;
  *qq=q;
  *size1=len1;
  *size2=len2;
  *kanji=kmode;
}

ptr s_prefix_length(int narg,ptr env)
{
  int len1,len2,kmode,s1;
  unsigned char *p,*q;

   set_string2(narg,&p,&q, &len1,&len2,&kmode);
 /**** printf("end1 %d end2 %d len1 %d len2 %d\n",
        end1,end2,len1,len2); ***/
  if(nocase)
      s1=prefix_cmp_len_ci(p,q,len1,len2);
   else
      s1=prefix_cmp_len(p,q,len1,len2);
  if(kmode){
    s1=bindex_jindex(p,s1,len1);
   }
  return MAKESINT(s1);
}

ptr s_suffix_length(int narg,ptr env)
{
  int len1,len2,kmode;
  int s1;
  unsigned char *p,*q;

   set_string2(narg,&p,&q, &len1,&len2,&kmode);
 /**** printf("end1 %d end2 %d len1 %d len2 %d\n",
        end1,end2,len1,len2); ***/
  if(nocase)
     s1=suffix_cmp_len_ci(p,q,len1,len2);
   else
     s1=suffix_cmp_len(p,q,len1,len2);
  if(kmode&&(s1>0)){
    s1=bindex_jindex(p,len1-s1,len1);
    s1=bindex_jindex(p,len1,len1)-s1;
   }
  return MAKESINT(s1);
}

ptr s_prefix_length_ci(int narg,ptr env)
{
  int len1,len2,kmode,s1;
  unsigned char *p,*q;



   set_string2(narg,&p,&q, &len1,&len2,&kmode);
 /**** printf("end1 %d end2 %d len1 %d len2 %d\n",
        end1,end2,len1,len2); ***/
  s1=prefix_cmp_len_ci(p,q,len1,len2);
  if(kmode){
    s1=bindex_jindex(p,s1,len1);
   }
  return MAKESINT(s1);
}



ptr s_suffix_length_ci(int narg,ptr env)
{
  int len1,len2,kmode;
  int s1;
  unsigned char *p,*q;



   set_string2(narg,&p,&q, &len1,&len2,&kmode);
```

```
 /**** printf("end1 %d end2 %d len1 %d len2 %d\n",
        end1,end2,len1,len2); ***/
  s1=suffix_cmp_len_ci(p,q,len1,len2);
  if(kmode&&(s1>0)){
    s1=bindex_jindex(p,len1-s1,len1);
    s1=bindex_jindex(p,len1,len1)-s1;
   }
  return MAKESINT(s1);
}

ptr s_prefix(int narg,ptr env)
{
  int len1,len2,kmode;
  int s1;
  unsigned char *p,*q;



   set_string2(narg,&p,&q, &len1,&len2,&kmode);
 /**** printf("end1 %d end2 %d len1 %d len2 %d\n",
        end1,end2,len1,len2); ***/
  if(nocase)
     s1=prefix_cmp_len_ci(p,q,len1,len2);
  else
     s1=prefix_cmp_len(p,q,len1,len2);
  return PRED(s1==len1);
}

ptr s_suffix(int narg,ptr env)
{
  int len1,len2,kmode;
  int s1;
  unsigned char *p,*q;



   set_string2(narg,&p,&q, &len1,&len2,&kmode);
 /**** printf("end1 %d end2 %d len1 %d len2 %d\n",
        end1,end2,len1,len2); ***/
  if(nocase)
       s1=suffix_cmp_len_ci(p,q,len1,len2);
  else
       s1=suffix_cmp_len(p,q,len1,len2);
  return PRED(s1==len1);
}
ptr s_prefix_ci(int narg,ptr env)
{
  int len1,len2,kmode;
  int s1;
  unsigned char *p,*q;



   set_string2(narg,&p,&q, &len1,&len2,&kmode);
 /**** printf("end1 %d end2 %d len1 %d len2 %d\n",
        end1,end2,len1,len2); ***/
  s1=prefix_cmp_len_ci(p,q,len1,len2);
  return PRED(s1==len1);
}



ptr s_suffix_ci(int narg,ptr env)
{
  int len1,len2,kmode;
  int s1;
  unsigned char *p,*q;



   set_string2(narg,&p,&q, &len1,&len2,&kmode);
 /**** printf("end1 %d end2 %d len1 %d len2 %d\n",
        end1,end2,len1,len2); ***/
  s1=suffix_cmp_len_ci(p,q,len1,len2);
  return PRED(s1==len1);
}
```