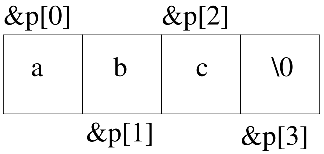
Strings

R. Inkulu
http://www.iitg.ac.in/rinkulu/

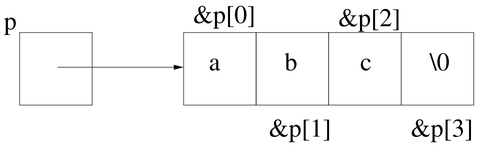
Array initialized with a string



```
   char p[] = "abc"; int i;
      //a (mutable) array of 4*sizeof(char) is created,
      //and initialized
   for (i=0; i<4; i++)
       printf("%c, ", p[i]);  //prints a, b, c,   ,
       //note the null character in the last entry
```

- "abc" in the above code is stored in the data segment; it is mutable (p is an array of 'char's)
- null character helps in finding the end of a string

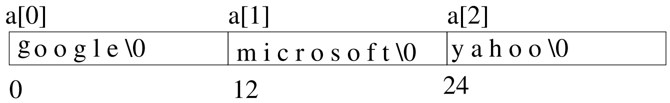
A pointer pointing to a constant string



```
   const char *p = "abc";
      //a non-mutable array of 4*sizeof(char) is created,
      //and initialized
   for (int i=0; i<4; i++)
       printf("%c, ", p[i]);  //prints a, b, c,  ,
       //again, note the null character at the end
```

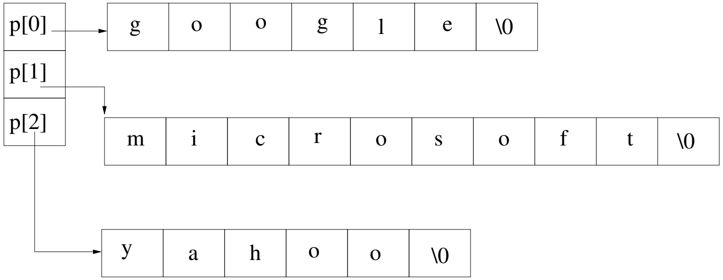
- "abc" in the above code is stored in the process text area; hence, is not mutable (p points to an array of 'const char's)

Multi-dimensional arrays



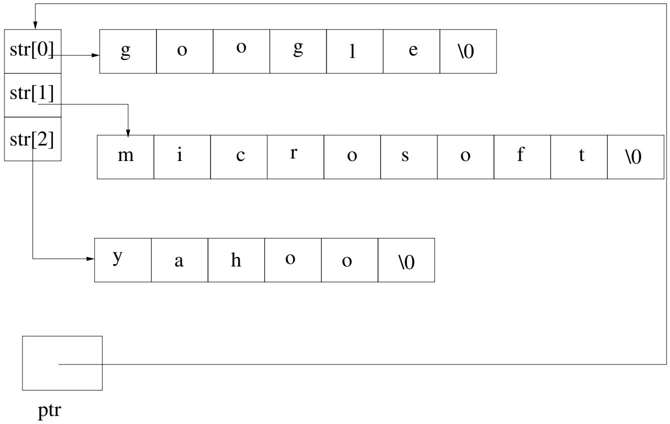
```
const char a[][12] = {"google", "microsoft", "yahoo"};

printf("%c, %c, %c\n", a[0][0], *(a[1]+3), *a[0]);
   //prints g, r, g

printf("%s, %s\n", a[0], a[1]+3);
   //prints google, rosoft

printf("%p, %p, %p\n", &a[0]+2, &a[2], a+2);
   //prints identical values

printf("%d, %d, %d\n",
       sizeof(a), sizeof(a[2]), strlen(a[2]));
   //prints 36, 12, 5
```

Array of pointers to strings



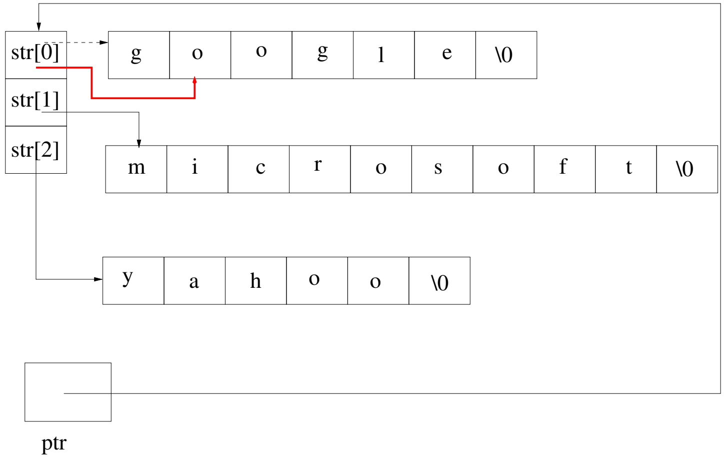
```
const char *p[] = {"google", "microsoft", "yahoo"};

printf("%s, %c, %c\n", p[0], p[1][2], *(p[2]+3));
    //prints google, c, o

printf("%d, %d, %d\n",
    sizeof(p), sizeof(p[2]), strlen(p[2]));
    //prints 12, 4, 5
```

Unary operators and pointers



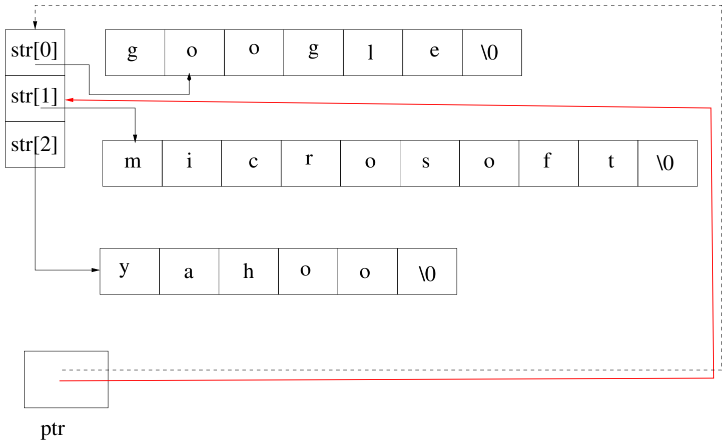
```
const char *str[] = {"google", "microsoft", "yahoo"};
const char **ptr = str;
printf("%s, %s, %s \n", *ptr, *(ptr+1), *(ptr+2));
    //prints google, microsoft, yahoo
```

Unary operators and pointers (cont)



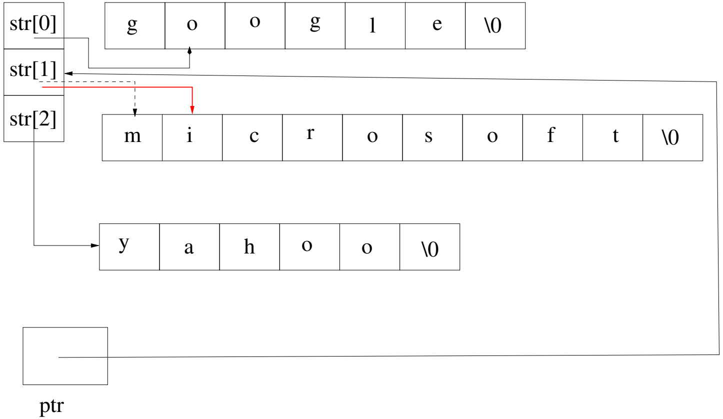
```
printf("%s, ", ++*ptr);    //prints oogle
```

Unary operators and pointers (cont)



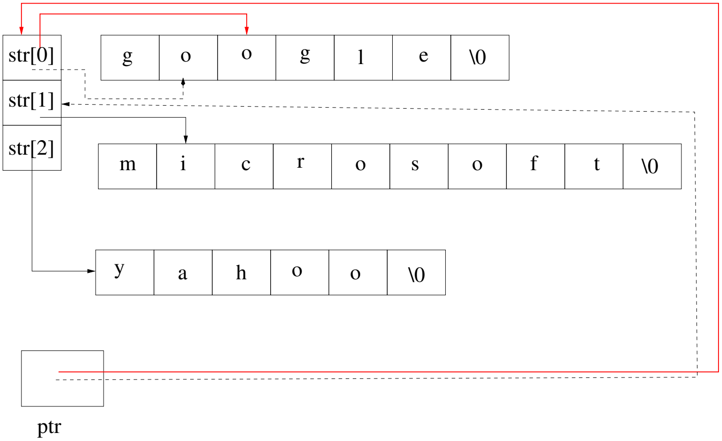
```
printf("%s, ", *ptr++);    //prints oogle
```

Unary operators and pointers (cont)



```
  printf("%s, ", ++*ptr);   //prints icrosoft
```

Unary operators and pointers (cont)



```
  printf("%s, ", ++*--ptr);   //prints ogle
```

strlen function definition

```
size_t strlen(const char *s) {
   size_t len = 0;
   while (*s++ != '\0') ++len;
   return len;
}

int main(void) {
   char a[] = "abcd";
   size_t len = strlen(a);
   printf("%d, \n", len);          //prints 4
   return 0;
}
```

- again observe 'const char *' vs 'char *' declarations

strcpy function definition

```
char *strcpy(char *to, const char *from) {
   char *tmp = to;
   while (*to++ = *from++);
   return tmp;
}
int func(char *a) {
   char *b;
   ...     //modified the contents of a somehow
   b = (char*)malloc((strlen(a)+1)*sizeof(char));
   strcpy(b, a);
   printf("%s, %s\n", a, b);
      //prints the same string twice
   ...
   free(b);
   ...  }
```

- heap memory facilitates to avoid allocating MAX-sized array on stack

strcmp function definition

```
int strcmp(const char *s, const char *t) {
   for (; *s == *t; s++, t++)
       if (*s == '\0')
          return 0;  //return 0 if s and t are same
   return *s - *t;
     //return < 0 if string pointed by s precedes t in
     //lexicographic ordering
     //return > 0 if string pointed by t precedes s
}

int main(void) {
   char a[] = "abcd";
   char *b = "abcde";
   if (!strcmp(b, a))
      printf("same");  //does not get printed
   return 0;
}
```

Few useful functions

- char *strcpy(char *dest, const char *src)
- char *strncpy(char *dest, const char *src, size_t num)
- size_t strlen(const char *s)
- int strcmp(const char *str1, const char *str2)
- int strncmp(const char *str1, const char *str2, size_t num)
- char *strcat(char *dest, const char *src)
- char *strncat(char *dest, const char *src, size_t num)
- char *strdup(const char *s)

  — only function listed in this page that allocates memory for the user; user is responsible to free the allocated block

- char *strchr(const char *s, int c)
- char *strrchr(const char *s, int c)
- char *strstr(const char *s, const char *pattern)

Few useful functions (cont)

- void *memchr(const void *s, int c, size_t num)

  scans the initial *num* bytes of the memory region pointed by *s* for the first instance of *c*

- void *memset(void *s, int c, size_t num)

  sets the first *num* bytes of the block of memory pointed by *s* to the value *c*

- int system(const char *command)

  returns status code of the called command

Few useful functions (cont)

- int atoi(const char *s)
- long atol(const char *s)
- double atof(const char *s)

distinguish text and binary forms of a number

Using strtok function

```
char input[]="CS101 is an interesting course!haha!";
char delim[] = " !\t\n";
char *token;

char* token = strtok(input, delim);
while (token != NULL) {
    printf("%s\n", token);
    token = strtok(NULL, delim);
 }
```

- prototype: *char *strtok(char *s, const char *t)*
- tokens are returned one after the other in order, one per invocation; no need to free any memory!

strtok function definition

```
char* strtok(char* str, const char* delim) {
        static char* tokenptr = NULL;
        static char delimtmp;
        int delimlen = strlen(delim), i;

        if (str != NULL)
                tokenptr = str;
        else
                *tokenptr++ = delimtmp;

        char *tmptoken = tokenptr;
        while (*tokenptr != '\0') {
                for (i = 0; i < delimlen; i++) {
                        if (*tokenptr == delim[i]) {
                                delimtmp = delim[i];
                                *tokenptr = '\0';
                                return tmptoken; }
                }
                ++tokenptr;
        }
        return NULL;
}
```

homework: there are multiple bugs in the deliberate simple code mentioned above . . . play with the code and fix all of them

Using strtod function

```
char sMilkPrice[] = "18.50 2.38";
char *p = NULL;  double price, litres;
price = strtod(sMilkPrice, &p);
litres = strtod(p, NULL);
printf("Milk cost per litre: %lf", price/litres);
   //prints 7.773109
```

- prototype: *double strtod(const char *s, char **endp)*

  stores a pointer to any unconverted suffix in *endp unless endp is NULL

strtod function definition

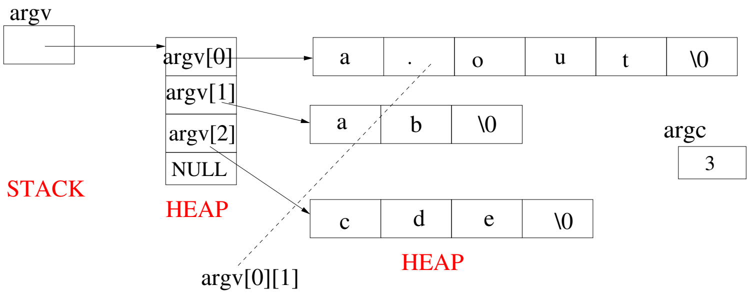
prototype: *double strtod(const char *s, char **endp)*

— high-level description of the same is given

Few more useful functions (cont)

- unsigned long strtoul(const char *s, char **endp)
- long strtol(const char *s, char **endp)

Memory-layout for command-line arguments



```
int main(int argc, char *argv[]) {
   //to remind, 'char *argv[]' gets translated
   //to 'char **argv'
   while (--argc > 0)
      printf("%s ", *++argv);
   //prints 'ab cde' when 'a.out ab cde' is executed
}
```