CSC552 – Advanced UNIX Programming

UNIX Fundamentals

Dr. L. Frye
Kutztown University

UNIX Introduction

- Flavors of UNIX
  - BSD
  - System V
- Terminology
  - Program
  - Process
  - Thread

Programs and Threads

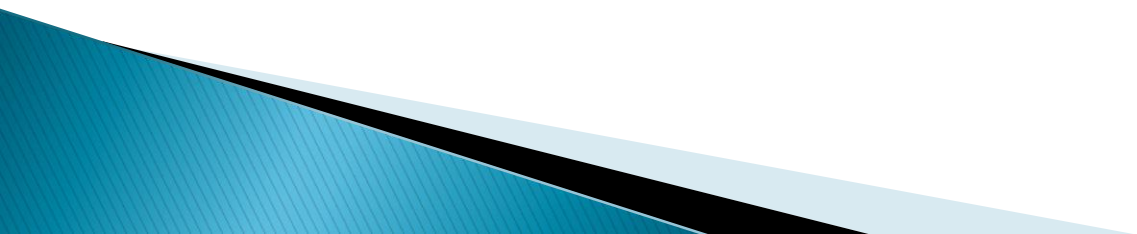
- Sequential program
- Concurrent program

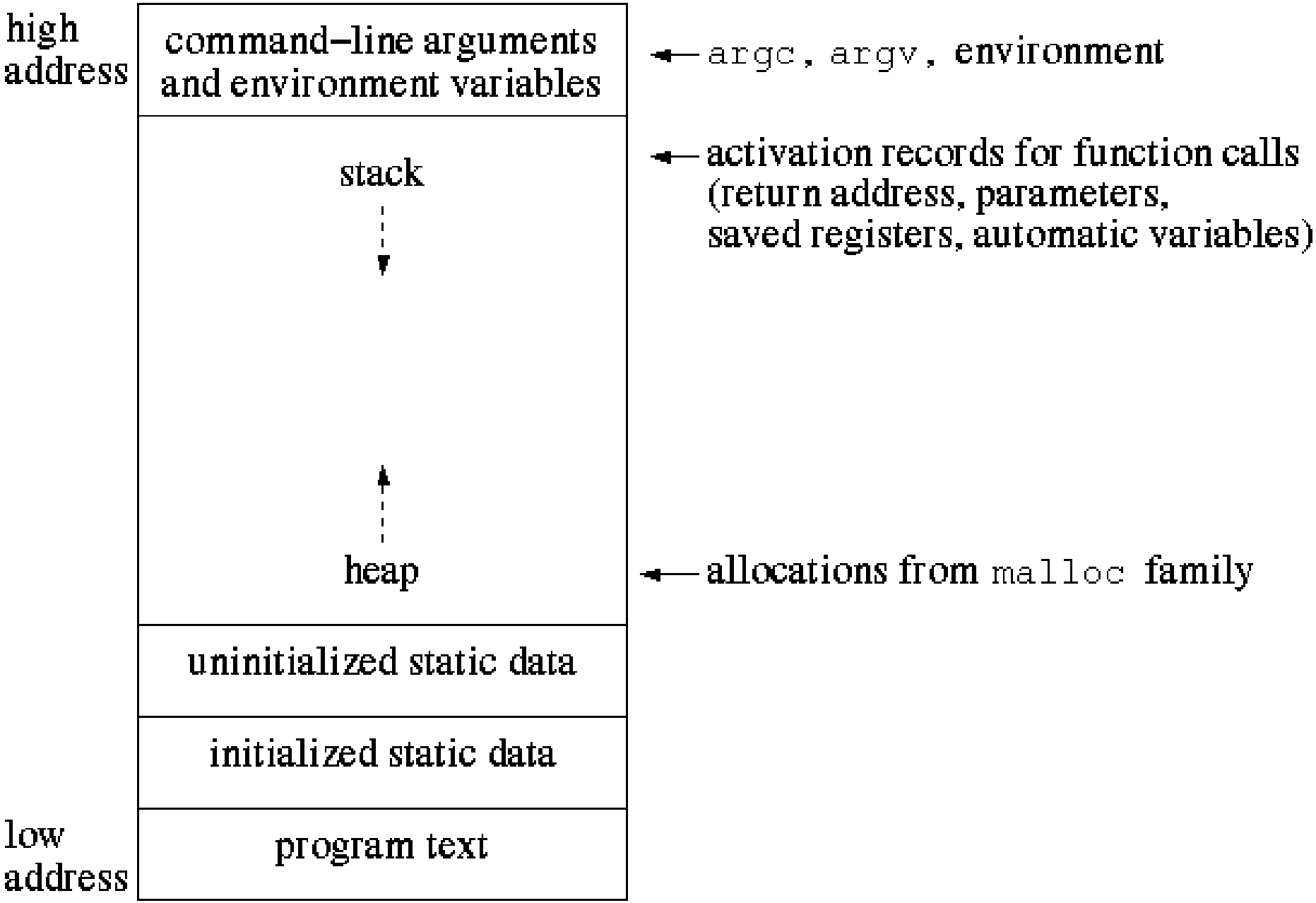
- What are the advantages of having multiple threads?

- Disadvantages?

Static Variables

- What are static variables?
- How are they used?




high address
command-line arguments and environment variables
argc, argv, environment
stack
activation records for function calls (return address, parameters, saved registers, automatic variables)
heap
allocations from malloc family
uninitialized static data
initialized static data
program text
low address

Handling Errors

- Handle all errors

- errno
- perror
- strerror

- Example programs
  - fundamentals/errno.c
  - fundamentals/perror.c

Re-executing Functions

- int fildes, error;
  while (((error = close(fildes)) == -1) &&
  (errno == EINTR)) ;
  if (error == -1)
  perror("Failed to close the file");

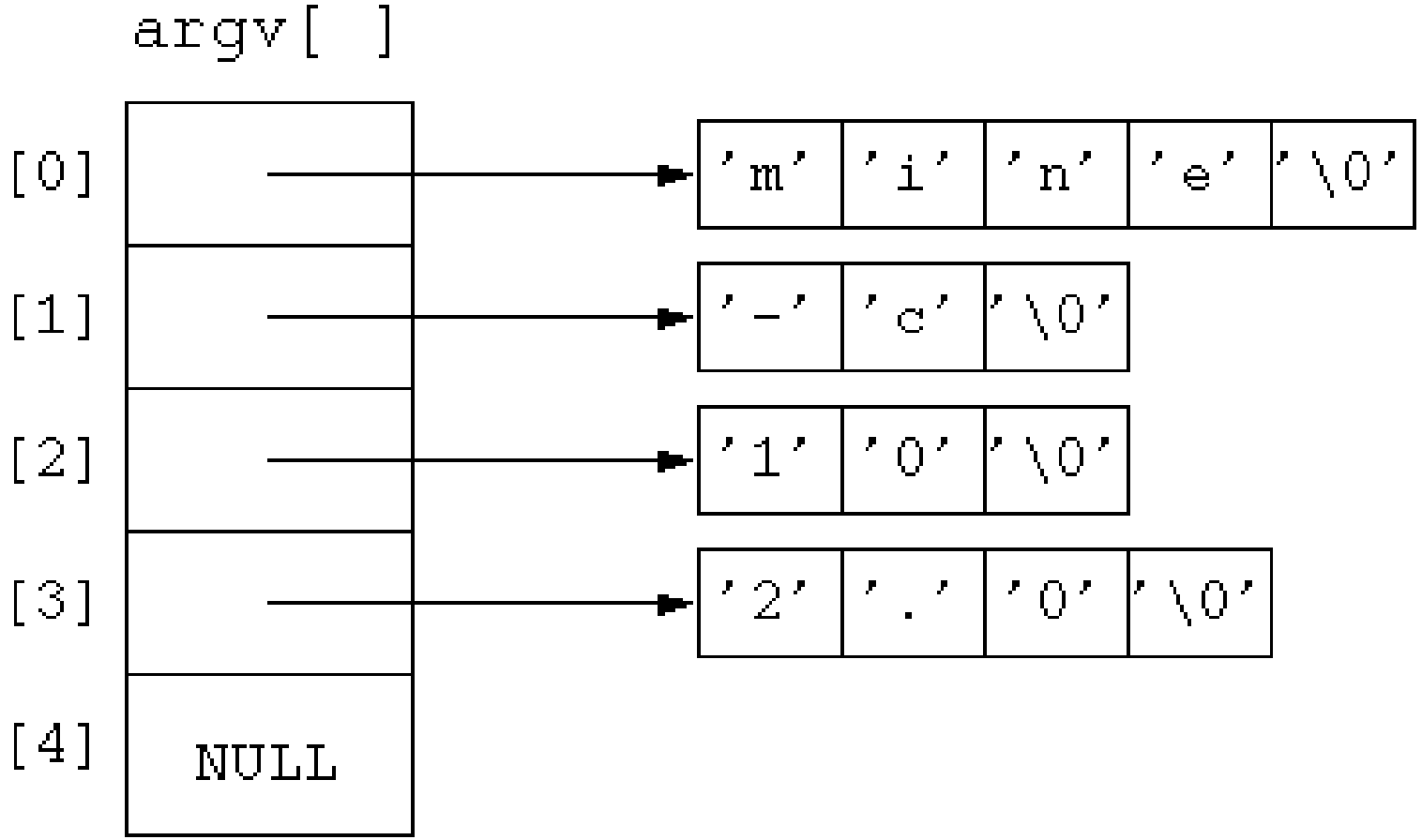
- Restart Library

- #include "restart.h"
  int fildes;
  if (r_close(fildes == -1) {
  perror("Failed to close the file");

Writing Functions

- Always return a value
- No unexpected changes to process state persisting beyond function
- Release all hidden resources

Argument Arrays



makeargv

- char **makeargv(char *s);

- int i;
  char **myargv;
  char mytest[] = “this is a test”;

  if ((myargv = makeargv(mytest)) == NULL)
      fprintf(stderr, “Failed to construct an
                          argument array\n”);
  else
      for (i = 0; myargv[i] != NULL; i++)
                      printf(“%d: %s\n”, i, myargv[i]);

makeargv

- int makeargv(char *s, char ***argvp);
-
- int i;
  char **myargv;
  char mytest[] = “This is a test”;
  int numtokens;

  if ((numtokens = makeargv(mytest, &myargv)) == -1)
      fprintf(stderr, “Failed to construct an argument
                                  array\n”);
  else
      for (i = 0; i < numtokens; i++)
                  printf(“%d: %s\n”, i, myargv[i]);

Thread-safe Functions

- fundamentals/wordaveragebad.c
- Why doesn't this program work correctly??
- fundamentals/wordaverage.c

Process Environment

- name = value
- sh and bash: set; export
- csh; setenv

- UNIX command → env

- fundamentals/printpath.c
- fundamentals/printenv.c

Programming Aids

- **man, apropos**
- **nohup** – ingnore hangup signal
- **lint** – find errors &inconstancies in C programs
- **truss** – trace program execution
- **prof** – show program profile
- **gprof** – Solaris

Libraries

- Static (archives)
- Shared (dynamic)

- What is the difference?

- Static Library Commands
  - ar
  - ranlib

- Compilation

Shared Libraries

- Position Independent Code (PIC)
  - gcc: -fpic
- Compilation
  - gcc: -shared
- LD_LIBRARY_PATH

Assignment – Shared Library

- Add functions in sockutils.c
- Create both a static and shared library
  - Call the library *libutils*

- Turn in a script containing the output of the following commands:
  - gcc for object file → gcc –c sockutils.c
  - ar → ar rc libutils.a sockutils.o
  - ranlib → ranlib libutils.a
  - gcc for shared library → gcc –fpic –c sockutils.c
    gcc –shared –o libutils.so sockutils.o
  - ls –l libutils.a libutils.so