Command Line Arguments

- Recall bash will sent arguments to commands
  - e.g. **grep –iw horse carol.txt**
  - the bash shell sends to the grep command the arguments '-iw' 'horse' 'carol.txt'
- The shell will also send arguments to your C programs
- **But how do we access these arguments?**

Accessing command line arguments

- Command-line arguments to a program are given to the program as an array of (pointers to) strings

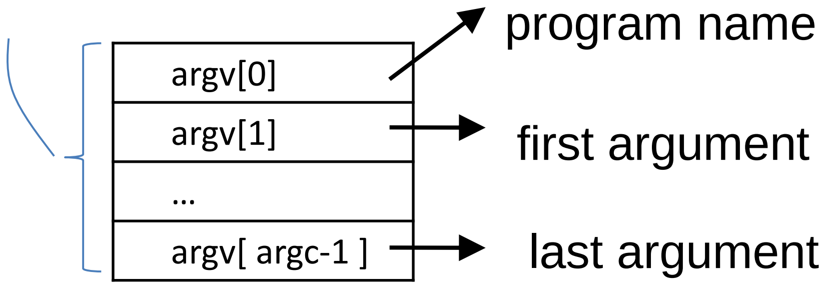
  char *argv[]

- No. of arguments (= size of this array) provided separately

  int argc

- Accessed within the program as

  int main(int argc, char *argv[])



argv[0]

argv[1]

…

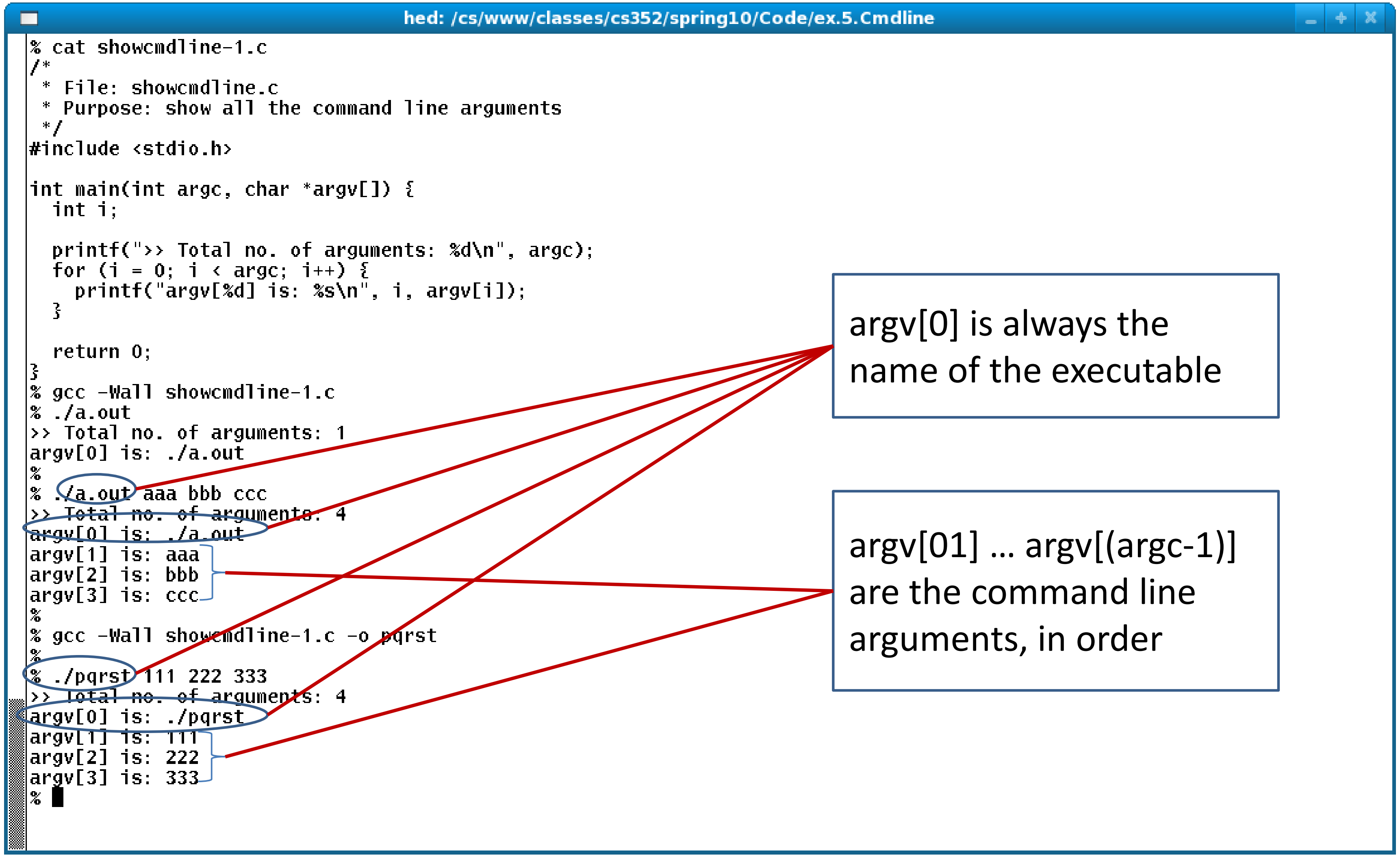
argv[argc-1]

program name

first argument

last argument

Example 1

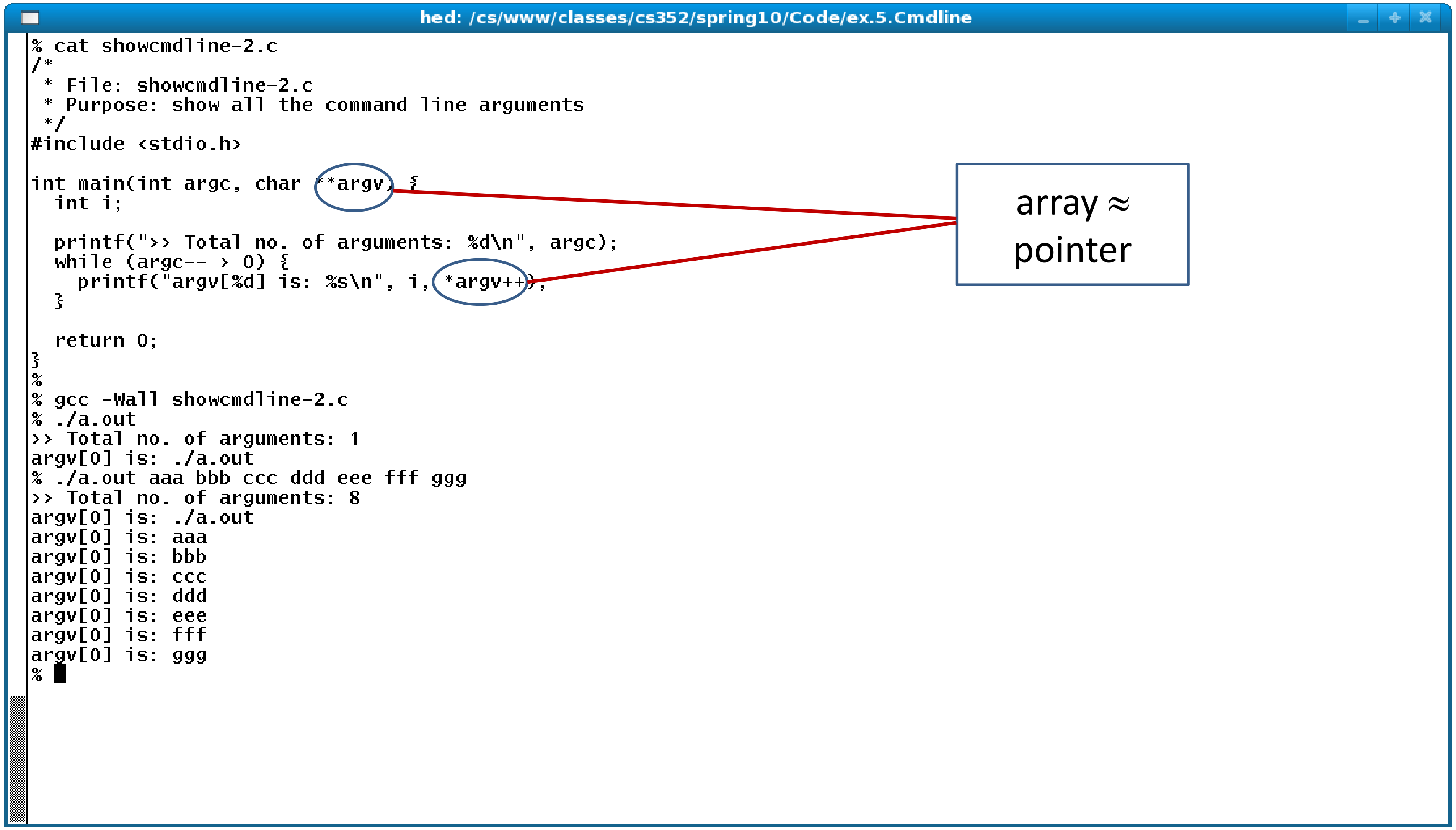


```
% cat showcmdline-1.c
/*
 * File: showcmdline.c
 * Purpose: show all the command line arguments
 */
#include <stdio.h>

int main(int argc, char *argv[]) {
  int i;

  printf(">> Total no. of arguments: %d\n", argc);
  for (i = 0; i < argc; i++) {
    printf("argv[%d] is: %s\n", i, argv[i]);
  }

  return 0;
}
% gcc -Wall showcmdline-1.c
% ./a.out
>> Total no. of arguments: 1
argv[0] is: ./a.out
%
% ./a.out aaa bbb ccc
>> Total no. of arguments: 4
argv[0] is: ./a.out
argv[1] is: aaa
argv[2] is: bbb
argv[3] is: ccc
%
% gcc -Wall showcmdline-1.c -o pqrst
%
% ./pqrst 111 222 333
>> Total no. of arguments: 4
argv[0] is: ./pqrst
argv[1] is: 111
argv[2] is: 222
argv[3] is: 333
% 
```

argv[0] is always the name of the executable

argv[01] … argv[(argc-1)] are the command line arguments, in order

Example 2



```
% cat showcmdline-2.c
/*
 * File: showcmdline-2.c
 * Purpose: show all the command line arguments
 */
#include <stdio.h>

int main(int argc, char **argv) {
  int i;

  printf(">> Total no. of arguments: %d\n", argc);
  while (argc-- > 0) {
    printf("argv[%d] is: %s\n", i, *argv++);
  }

  return 0;
}
%
% gcc -Wall showcmdline-2.c
% ./a.out
>> Total no. of arguments: 1
argv[0] is: ./a.out
% ./a.out aaa bbb ccc ddd eee fff ggg
>> Total no. of arguments: 8
argv[0] is: ./a.out
argv[0] is: aaa
argv[0] is: bbb
argv[0] is: ccc
argv[0] is: ddd
argv[0] is: eee
argv[0] is: fff
argv[0] is: ggg
% 
```

array ≈ pointer