Compilation and Linking under Unix

Computer System

High-level language program (in C)

```
swap(int v[], int k)
{int temp;
   temp = v[k];
   v[k] = v[k+1];
   v[k+1] = temp;
}
```

C Compiler

Assembly language program (for MIPS)

```
swap:
      muli $2, $5,4
      add  $2, $4,$2
      lw   $15, 0($2)
      lw   $16, 4($2)
      sw   $16, 0($2)
      sw   $15, 4($2)
      jr   $31
```

Assembler

Binary machine language program (for MIPS)

```
00000000101000010000000000011000
00000000100011100001100000100001
10001100011000100000000000000000
10001100111100100000000000000100
10101100111100100000000000000000
10101100011000100000000000000100
00000011111000000000000000001000
```

Computer System

- Each type of CPU executes only a particular *machine language*
- A program must be translated into machine language before it can be executed
- A *compiler* is a software tool which translates *source code* into a specific target language
- Often, that target language is the machine language for a particular CPU type

Program Development

- The mechanics of developing a program include several activities
  - writing the program in a specific programming language (such as C and Java)
  - translating the program into a form that the computer can execute
  - investigating and fixing various types of errors that can occur
- Software tools can be used to help with all parts of this process

Single-module Programs

- Let's examine a C program that performs a simple task: reversing a string.
- We will learn how to write, compile, link, and execute a program that solves the problem using a single source file.
  - It's better to split a large program up into several independent modules. (will be discussed later)
- A source code listing of the first version of the reverse program is next presented.

```
/*  reverse.c  */
#include <stdio.h>
#include <string.h>
/*  Function prototype  */
void reverse (char before[], char after[]);

/*********************************************/
int main()
{
  char str[100];   /* buffer to hold reversed string */
  reverse("cat",str);  /* reverse the string "cat"  */
  printf ("reverse("cat") = %s\n", str);
  reverse("noon",str);
  printf ("reverse("noon") = %s\n", str);
}

/*********************************************/
void reverse (char before[], char after[])
{
  int i,j,len;

  len = strlen(before);
  i=0;
  for (j=len-1; j>=0; j--)
  {
    after[i] = before[j];
    i++;
  }
  after[len] = '\0';
}
```

C Program Development

- First we need to type the program source code into a file called source file (or source program file).
  - Suppose that we call this file “reverse.c”
- The format of the source file should be a *text file* since it contains text characters
- A text file in Unix can be created using an Unix editor such as **pico** (**nano** in Linux), **emacs**, **pico**, **vim**.
- Suppose that we first create a directory (folder) called “reverse” under my home directory to prepare the development of the program.

C Compilers

- Until recently, a C compiler was a standard component in UNIX, especially UNIX versions that came with source code.
  - Depending on your needs, you should check on this in any version of UNIX you are considering using.
- Even if your version of UNIX no longer ships a C compiler, you have an alternative, thanks to the GNU Project: GNU C (gcc) and GNU C++ (g++) are freely available C and C++ compilers, respectively
  - GNU Compiler Collection web site, http://www.gnu.org/software/gcc/gcc.html

Compiling a C program

- Compile the C program with the **gcc** utility.
- By default, gcc creates an *executable* file called “a.out” in the current directory.
- The format of executable file is called *binary* file which contains binary bits.
  - Therefore, we cannot use the Unix utilities *cat* or *more* to display the content on the screen.
- To run the program, type “./a.out”. Any errors that are encountered are sent to the standard error channel, which is connected by default to your terminal’s screen.

Compiling a C program (con't)

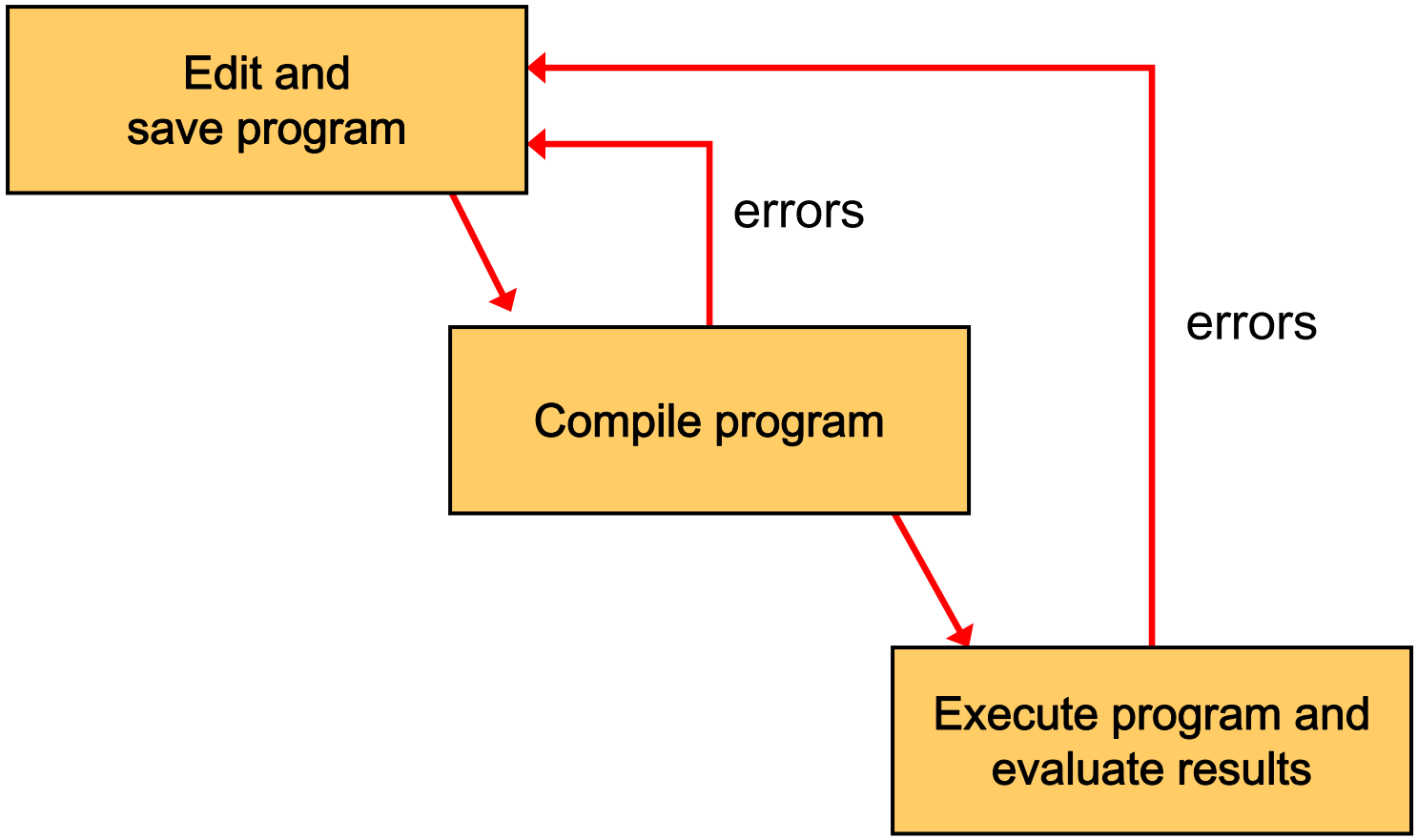
- Here's what happened when I compiled my program:

```
sepc92:> mkdir reverse                    ... create subdirectory for source code
sepc92:> cd reverse
sepc92:> nano reverse.c                   ... create the file reverse.c using pico


sepc92:> gcc reverse.c                    ... compile source.
reverse.c: In function 'main':
reverse.c:12: parse error before 'cat'
reverse.c:14: parse error before 'noon'
sepc92:>
```

- As you can see, gcc found a number of compile-time errors, listed together with their causes as follows:
  - The errors on lines 12 and 14 were due to an inappropriate use of double quotes within double quotes.

Basic Program Development



Compiling a C program (con't)

- If there are so many compilation errors that it cannot fit into one screen. One way is to compile by the following command:

  gcc reverse.c |& more

  which means that the compilation errors will be displayed screen by screen.
  - It will display the first screen of errors and wait. After the user examines the errors, he/she can choose to display the next screen of errors by hitting RETURN or quit by hitting Control-C.
- The corrected version of the reverse program is given in the next page.

```
/*  reverse.c  */
#include <stdio.h>
#include <string.h>
/*  Function prototype  */
void reverse (char before[], char after[]);

/*********************************************/
int main()
{
  char str[100];   /* buffer to hold reversed string */
  reverse("cat",str);  /* reverse the string "cat"  */
  printf ("reverse(\"cat\") = %s\n", str);
  reverse("noon",str);
  printf ("reverse(\"noon\") = %s\n", str);
}

/*********************************************/
void reverse (char before[], char after[])
{
  int i,j,len;

  len = strlen(before);
  i=0;
  for (j=len-1; j>=0; j--)
  {
    after[i] = before[j];
    i++;
  }
  after[len] = '\0';
}
```

Compiling a C program (con't)

- If there are many compilation errors, it is a good strategy to debug the first few errors since some remaining compilation errors may not be actual errors.
  - They exist due to the limitations of the compiler. Hence, fixing earlier errors will usually result in fewer errors encountered in the debugging process.
  - e.g. a semicolon is mistakenly added after the function heading line of reverse would produce a lot of compilation errors.

Compiling a C program (con't)

- **Running a C Program**

  After compiling the second version of "reverse.c", I ran it by typing the name of the executable file, "a.out". As you can see, the answers were correct:

```
sepc92:> gcc  reverse.c                          ...compile source.
sepc92:> ls  –l  reverse.c  a.out                ...list file information.
-rwxr-xr-x 1 glass          24576 Jan5 16:16     a.out*
-rw----r--  1 glass          439     Jan 5 16:15   reverse.c
sepc92:> ./a.out                                 ... run program
reverse ("cat") = tac
reverse ("noon") = noon
sepc92:>
```

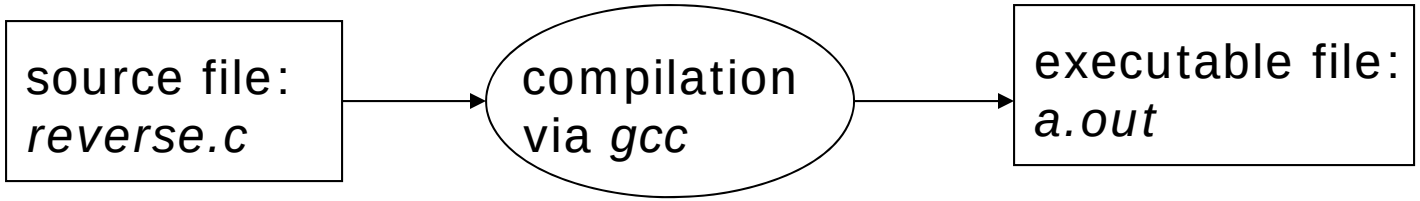
Compiling a C program (con't)

- **Overriding the Default Executable Name**
  - The name of the default executable file, "a.out", is rather cryptic, and an "a.out" file produced by a subsequent compilation would overwrite the one that I just produced.
  - To avoid both problems, it's best to use the -o option with gcc**,** which allows you to specify the name of the executable file that you wish to create:

```
sepc92:> gcc  -o reverse  reverse.c        ... name the executable "reverse"
sepc92:> ls  -l  reverse
-rwxr-xr-x     1 glass        24576  Jan 5 16:19   reverse*
sepc92:> ./reverse                          ... run the executable "reverse"
reverse ("cat") = tac
reverse ("noon") = noon
sepc92:>
```

Summary of *gcc* utility

gcc reverse.c



gcc –o reverse reverse.c