Unix System Programming
Files and Directories
Process

Chandravva Hebbi
Department of Computer Science and Engineering
chandravvahebbi@pes.edu

Unix System Programming

Files and Directories

Process

Chandravva Hebbi

Department of Computer Science and Engineering

UNIX SYSTEM PROGRAMMING

Topics to be Covered

- Reading Directories
- Programming Examples on file access permissions, access,
- Programming Examples on reading directories
- Chdir, getcwd
- Process Environment
- Memory Layout of a C program
- Exit functions

Reading Directories

- Directory content can be read by anyone having read permission
  - Earlier systems, such as Version 7, had a simple structure
  - each directory entry was 16 bytes, with 14 bytes for the filename and 2 bytes for the i-node number.

```
#include <dirent.h>
DIR *opendir(const char *pathname);
                                Returns: pointer if OK, NULL on error

struct dirent *readdir(DIR *dp);
                Returns: pointer if OK, NULL at end of directory or error
```

Reading Directories

void rewinddir(DIR *dp);

int closedir(DIR *dp);

Returns: 0 if OK, -1 on error

long telldir(DIR *dp);

Returns: current location in directory associated with *dp*

void seekdir(DIR *dp, long loc);

Struct dirent

```
struct dirent {
ino_t d_ino; /* i-node number *// not defined in POSIX.1
char d_name[NAME_MAX + 1]; /* null-terminated filename */
}
```

Change directory

- Change the Directory

int chdir(const char *_pathname_);

- Get the current working directory

```
char *getcwd(char *buf, size_t size);
```

Returns: *buf* if OK, `NULL` on error

- **main Function**
- The prototype for the main function is

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

- When a C program is executed by the kernel by one of the exec functions.
- special start-up routine is called before the main function is called.
- The executable program file specifies this routine as the starting address for the program;
- This is set up by the link editor when it is invoked by the C compiler.
- This start-up routine takes values from the kernel
- The command-line arguments and the environment and sets things up so that the main function is called

Process Termination

There are eight ways for a process to terminate

Normal termination occurs in five ways:

1. Return from main

2. Calling exit

3. Calling _exit or _Exit

4. Return of the last thread from its start routine

5. Calling pthread_exit from the last thread

Abnormal termination occurs in three ways:

6. Calling `abort`
7. Receipt of a signal
8. Response of the last thread to a cancellation request

Exit Functions

- Three functions terminate a program normally: _exit and _Exit, which return to the kernel immediately
- exit, which performs certain cleanup processing and then returns to the kernel.

```
#include <stdlib.h>
void exit(int status);
void _Exit(int status);
#include <unistd.h>
void _exit(int status);
```

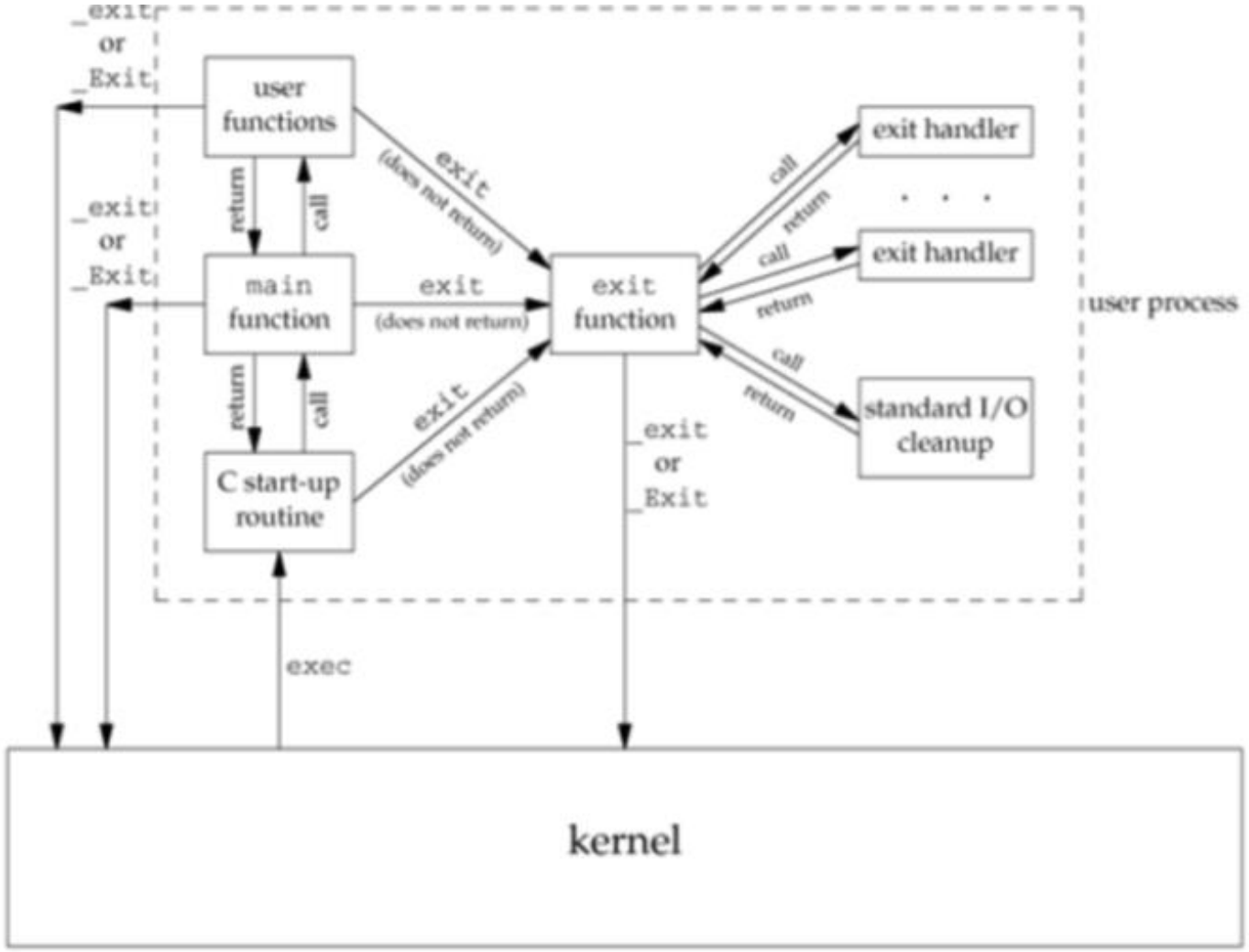
exit functions

- The exit function has always performed a clean shutdown of the standard I/O library
- The fclose function is called for all open streams.
- All three exit functions expect a single integer argument, which we call the exit status.

```
exit(0);
is the same as
return(0);
```

UNIX SYSTEM PROGRAMMING

Execution of C Program



THANK YOU

Chandravva Hebbi
Department of Computer Science and Engineering

chandravvahebbi@pes.edu