Error Handling

- Should *always* check return code of system calls
  - Not only for 5 points in your lab!
  - There are subtle ways that things can go wrong
  - Use the status info kernel provides us
- Approach in this class: Wrappers
- Different error handling styles
  - Unix-Style
  - Posix-Style
  - DNS-Style

Unix-Style Error Handling

- Special return value when encounter error (always –1)
- Set global variable **errno** to an error code
  - Indicates the cause of the error
- Use **strerror** function for text description of **errno**

```
void unix_error(char *msg)
{
  fprintf(stderr, “%s: %s\n“,
            msg, strerror(errno));
  exit(0);
}
… …
if ((pid = wait(NULL)) < 0)
  unix_error(“Error in wait”);
```

POSIX-Style Error Handling

- Return value only indicate success (0) or failure (nonzero)
- Useful results returned in function arguments

```
void posix_error(int code, char *msg)
{
  fprintf(stderr, "%s: %s\n",
                  msg, strerror(code));
  exit(0);
}
… …
if ((retcode = pthread_create(…)) != 0)
  posix_error(retcode, "Error in pthread");
```

DNS-Style Error Handling

- Return a NULL pointer on failure
- Set the global `h_errno` variable

```
void dns_error(char *msg)
{
  fprintf(stderr, “%s: DNS error %d\n“,
                msg, h_errno);
  exit(0);
}
… …
if ((p = gethostbyname(name)) == NULL)
  dns_error(“Error in gethostbyname”);
```

Example: Wrappers

```
void Kill (pid_t pid, int signum)
{
  int rc;
  if((rc = kill(pid, signum)) <0)
      unix_error("Kill error");
}
```

- Appendix B: csapp.c and csapp.h
- Unix-Style, for kill function
- Behaves exactly like the base function if no error
- Prints informative message and *terminates* the process

Not All Errors are Fatal

- Wrappers are not always the correct path
  - Treat all information as fatal errors
  - Terminate the program with **`exit()`**
  - Sometimes an error is not fatal

```
void sigchld_handler(int signum)
{
  pid_t pid;

  while((pid = waitpid(-1, NULL, 0)) > 0)
    printf("Reaped %d\n", (int)pid);

  if(errno != ECHILD)
    unix_error("waitpid error");
}
```