2015/12/21

システムプログラミング演習6

学籍番号：201420694
氏名：星 遼平

1　演習 6-1

プログラム

```
#include <stdlib.h>
#include <stdio.h>
#include <unistd.h>
#include <errno.h>
#include <string.h>
#include <syslog.h>
#include <sys/types.h>
#include <sys/socket.h>
#include <netinet/in.h>

#include "logutil.h"

#define DEFAULT_SERVER_PORT 10000
#ifdef SOMAXCONN
#define LISTEN_BACKLOG SOMAXCONN
#else
#define LISTEN_BACKLOG 5
#endif

char *program_name = "sp6-server";
```

プログラム（続き）

```
int open_accepting_socket(int port) {
  struct sockaddr_in self_addr;
  socklen_t self_addr_size;
  int sock, sockopt;

  memset(&self_addr, 0, sizeof(self_addr));
  self_addr.sin_family = AF_INET;
  self_addr.sin_addr.s_addr = INADDR_ANY;
  self_addr.sin_port = htons(port);
  self_addr_size = sizeof(self_addr);
  sock = socket(PF_INET, SOCK_STREAM, 0);
  if (sock < 0)
    logutil_fatal("accepting socket: %d", errno);
  sockopt = 1;
  printf("test\n");
  if (setsockopt(sock, SOL_SOCKET, SO_REUSEADDR,
        &sockopt, sizeof(sockopt)) == -1)
    logutil_warning("SO_REUSEADDR: %d", errno);
  if (bind(sock, (struct sockaddr *)&self_addr, self_addr_size) < 0)
    logutil_fatal("bind accepting socket: %d", errno);
  if (listen(sock, LISTEN_BACKLOG) < 0)
    logutil_fatal("listen: %d", errno);
  return (sock);
}

void usage(void) {
  fprintf(stderr, "Usage: %s [option]\n", program_name);
  fprintf(stderr, "option:\n");
  fprintf(stderr, "\t-d\t\t\t\t... debug mode\n");
  fprintf(stderr, "\t-p <port>\n");
  exit(1);
}
```

プログラム（続き）

```
void main_loop(int self_addr) {
  int sock, n;
  struct sockaddr_in client_addr;
  socklen_t client_addr_size;
  char buf[1024];

  client_addr_size = sizeof(client_addr);
  if ((sock =
  accept(self_addr, (struct sockaddr *) &client_addr, &client_addr_size)) < 0) {
    logutil_fatal("deny access: %d", errno);
  }

  while (1) {
    read(sock, buf, 1024);

    if (buf[0] == EOF) {
      logutil_info("disconnected\n");
      break;
    }
    write(sock, buf, strlen(buf) + 1);
  }
  close(sock);
}

int main(int argc, char **argv) {
  char *port_number = NULL;
  int ch, sock, server_port = DEFAULT_SERVER_PORT;
  int debug_mode = 0;

  while ((ch = getopt(argc, argv, "dp:")) != -1) {
    switch (ch) {
      case 'd':
        debug_mode = 1;
        break;
      case 'p':
        port_number = optarg;
        break;
      case '?':
      default:
        usage();
    }
  }
  argc -= optind;
  argv += optind;
```

プログラム（続き）

```
  if (port_number != NULL)
    server_port = strtol(port_number, NULL, 0);

  /* server_port で listen し，socket descriptor を sock に代入 */
  sock = open_accepting_socket(server_port);

  if (!debug_mode) {
    logutil_syslog_open(program_name, LOG_PID, LOG_LOCAL0);
    daemon(0, 0);
  }

  /*
   * 無限ループで sock を accept し，accept したらそのクライアント用
   * のスレッドを作成しプロトコル処理を続ける.
   */
  main_loop(sock);

  /*NOTREACHED*/
  return (0);
}
```

実行結果

```
$ ./app -d
test

$ telnet localhost 10000
Trying 127.0.0.1...
Connected to localhost.
Escape character is '^]'.
^CConnection closed by foreign host.

$ ./app -d
test
disconnected
```

2　演習 6-2

プログラム

```
#include <stdlib.h>
#include <stdio.h>
#include <unistd.h>
#include <errno.h>
#include <string.h>
#include <syslog.h>
#include <sys/types.h>
#include <sys/socket.h>
#include <netinet/in.h>
#include <signal.h>
#include <pthread.h>

#include "logutil.h"

#define DEFAULT_SERVER_PORT 10000
#ifdef SOMAXCONN
#define LISTEN_BACKLOG SOMAXCONN
#else
#define LISTEN_BACKLOG 5
#endif

char *program_name = "sp6-server";

pthread_mutex_t m = PTHREAD_MUTEX_INITIALIZER;
sigset_t intset;
sigset_t termset;
int sigint = 0;
int sigterm = 0;
```

プログラム（続き）

```
int open_accepting_socket(int port) {
  struct sockaddr_in self_addr;
  socklen_t self_addr_size;
  int sock, sockopt;

  memset(&self_addr, 0, sizeof(self_addr));
  self_addr.sin_family = AF_INET;
  self_addr.sin_addr.s_addr = INADDR_ANY;
  self_addr.sin_port = htons(port);
  self_addr_size = sizeof(self_addr);
  sock = socket(PF_INET, SOCK_STREAM, 0);
  if (sock < 0)
    logutil_fatal("accepting socket: %d", errno);
  sockopt = 1;
  printf("test\n");
  if (setsockopt(sock, SOL_SOCKET, SO_REUSEADDR,
        &sockopt, sizeof(sockopt)) == -1)
    logutil_warning("SO_REUSEADDR: %d", errno);
  if (bind(sock, (struct sockaddr *)&self_addr, self_addr_size) < 0)
    logutil_fatal("bind accepting socket: %d", errno);
  if (listen(sock, LISTEN_BACKLOG) < 0)
    logutil_fatal("listen: %d", errno);
  return (sock);
}
```

プログラム（続き）

```
void usage(void) {
  fprintf(stderr, "Usage: %s [option]\n", program_name);
  fprintf(stderr, "option:\n");
  fprintf(stderr, "\t-d\t\t\t\t... debug mode\n");
  fprintf(stderr, "\t-p <port>\n");
  exit(1);
}

void *handle_int(void *arg) {
  int sig, err;

  err = sigwait(&intset, &sig);
  if (err || sig != SIGINT)
    abort();
  pthread_mutex_lock(&m);
  logutil_info("\nbye");
  sigint = 1;
  pthread_mutex_unlock(&m);
}

void *handle_term(void *arg) {
  int sig, err;

  err = sigwait(&termset, &sig);
  if (err || sig != SIGTERM)
    abort();
  pthread_mutex_lock(&m);
  logutil_info("bye\n");
  sigterm = 1;
  pthread_mutex_unlock(&m);
}
```

プログラム（続き）

```
void main_loop(int self_addr) {
  int sock, n;
  struct sockaddr_in client_addr;
  socklen_t client_addr_size;
  char buf[1024];

  client_addr_size = sizeof(client_addr);
  if ((sock =
    accept(self_addr, (struct sockaddr *) &client_addr, &client_addr_size)) < 0) {
    logutil_fatal("deny access: %d", errno);
  }

  while (1) {
    read(sock, buf, 1024);

    if (buf[0] == EOF) {
      logutil_info("disconnected\n");
      break;
    }

    if (sigint || sigterm) {
      logutil_info("disconnected\n");
      break;
    }

    write(sock, buf, strlen(buf) + 1);
  }
  close(sock);
}
```

プログラム（続き）

```
int main(int argc, char **argv) {
  char *port_number = NULL;
  int ch, sock, server_port = DEFAULT_SERVER_PORT;
  int debug_mode = 0;
  pthread_t t;

  while ((ch = getopt(argc, argv, "dp:")) != -1) {
    switch (ch) {
      case 'd':
        debug_mode = 1;
        break;
      case 'p':
        port_number = optarg;
        break;
      case '?':
      default:
        usage();
    }
  }
  argc -= optind;
  argv += optind;
```

プログラム（続き）

```
  if (port_number != NULL)
    server_port = strtol(port_number, NULL, 0);

  /* server_port で listen し，socket descriptor を sock に代入 */
  sock = open_accepting_socket(server_port);

  if (!debug_mode) {
    logutil_syslog_open(program_name, LOG_PID, LOG_LOCAL0);
    daemon(0, 0);
  }

  sigemptyset(&intset);
  sigemptyset(&termset);

  sigaddset(&intset, SIGINT);
  sigaddset(&termset, SIGTERM);

  pthread_sigmask(SIG_BLOCK, &intset, NULL);
  pthread_sigmask(SIG_BLOCK, &termset, NULL);

  pthread_create(&t, NULL, handle_int, NULL);
  pthread_create(&t, NULL, handle_term, NULL);

  /*
   * 無限ループで sock を accept し，accept したらそのクライアント用
   * のスレッドを作成しプロトコル処理を続ける．
   */
  main_loop(sock);

  /*NOTREACHED*/
  return (0);
}
```

実行結果（SIGINT の場合）

```
$ ./app -d
test

$ ./app -d
test
^C
bye
```

実行結果（SIGTERM の場合）

```
$ ./app -d
test

$ ps aux | grep app
vagrant   5754  0.0  0.0  22888   384 pts/0    Sl+  11:35   0:00 ./app -d

$ ./app -d
test
bye
```

3　考察

シグナルを利用することで Ctrl-C などの特別な入力や kill コマンドでプロセスを殺す際にプログラムを作成者が意図したとおりに通ささせることが可能であることが今回の演習を通して学ぶことができた．シグナルをうまく利用すれば，安全にプロセスを実行させられる．ただ，kill コマンドのオプションで強制的にプロセスを終了するものがあるが，シグナルをキャッチして強制オプションがつけられても殺せないようなプロセスを実行することが可能であるのかどうか気になった．もし，強制オプションがあっても殺せないようなプロセスを生成してしまったら，少々厄介であると感じた．

4　授業の感想

個人的に今回の授業内容は難易度が高いと感じました．また，前回の授業のようにちょっとしたサンプルプログラムを書いて実行してみると理解度も高まったのではないかと感じました．