Echoサーバのプログラム例

```
#include <errno.h>
#include <stdlib.h>
#include <stdio.h>
#include <string.h>
#include <unistd.h>
#include <sys/socket.h>          // ソケット関係ライブラリ
#include <netinet/in.h>          // インターネットソケットアドレスなど

void
fatal(char *msg)                 // エラーメッセージを出力し終了
{
        perror(msg);
        exit(EXIT_FAILURE);
}
```

```
int
open_accepting_socket(int port)       // 接続可能状態のソケットを作成
{
        struct sockaddr_in addr;
        int sock, sockopt;

        memset(&addr, 0, sizeof(addr));            // ゼロクリア
        addr.sin_family = AF_INET;                 // アドレスファミリはINET（インターネット）
        addr.sin_addr.s_addr = INADDR_ANY;         // クライアント側のIPアドレスは指定しない
        addr.sin_port = htons(port);               // ポート番号をネットワーク順序で指定

        sock = socket(PF_INET, SOCK_STREAM, 0);    // プロトコルファミリはインターネットで
        if (sock < 0)                              // ソケットを作成
                fatal("socket");
        sockopt = 1;
        if (setsockopt(sock, SOL_SOCKET, SO_REUSEADDR,   // すぐ再利用可能とするため
                        &sockopt, sizeof(sockopt)) == -1)        // ソケットオプションを指定
                perror("sockopt");
        if (bind(sock, (struct sockaddr *)&addr, sizeof(addr)) < 0)
                fatal("bind");
        if (listen(sock, SOMAXCONN) < 0)           //  指定したアドレスにバインドして
                fatal("listen");                   //  接続可能状態に

        return (sock);
}
```

```
#define MAXDATA 1024

void
echo_server(int client)                    // エコーサーバ
{                                          // クライアントのソケットディスクリプタを渡す
        char buf[MAXDATA];
        int rv;

        for (;;) {                         // えんえん繰り返す
                rv = read(client, buf, MAXDATA);    // クライアントからデータを受信
                if (rv == 0) /* EOF */              // EOFの場合
                        break;
                else if (rv < 0) {                  // エラーの場合
                        perror("read");
                        break;
                } else
                        write(client, buf, rv);     // 受信したデータをクライアントに送信
        }
        printf("Connection closed.¥n");
}
```

```
void
accepting_loop(int sock)                    // メインループ
{
        struct sockaddr_in client;
        socklen_t clientlen = sizeof(client);
        int client_sock;

        for (;;) {                                                    // クライアントの接続を待つ
                client_sock = accept(sock,                            // クライアントのソケットデスクリプタを
                   (struct sockaddr *)&client, &clientlen);           // 取得
                if (client_sock < 0)
                        if (errno != EINTR)                           // エラー処理
                                perror("accept");                     // 中断割込でなければ終了
                        else
                                continue;                             // 中断割込の場合続行
                else
                        echo_server(client_sock);                     // エコーサーバを呼ぶ
        }                                                             // 並行サーバにするためには
}                                                                     // fork() or pthread_create()を呼ぶ
```

メイン関数

```
int
main()
{
	int sock;

	sock = open_accepting_socket(7777);
	accepting_loop(sock);

	return (0);
}
```

実行の模様

サーバのコンパイル、実行

```
$ cc -Wall -O -o echod echod.c        // コンパイル
$ ./echod                             // echoサーバを実行
Connection closed.                    // クライアントが接続を切断
```

クライアントの実行

```
$ telnet localhost 7777               // tcp/7777でlistenしているechoサーバに接続
Trying 127.0.0.1...
Connected to localhost.
Escape character is '^]'.
input from client                     // 太字はクライアントからの入力
input from client                     // サーバからの返事
hello
hello
^]                                    // ^]を入力してエスケープ
telnet> Connection closed.            // Ctrl-Dで切断
```

改良点

- スレッドプールを利用した並行サーバに
- daemon(0, 0)を呼んでサーバをバックグラウンドで実行（デーモンに）
  - daemon(3)
- その場合、エラーメッセージはsyslogに
  - openlog(3), syslog(3), closelog(3)