

## 課題 No.11 の解答例

1. myshell に環境変数を追加するコマンド `setenv` , 削除するコマンド `unsetenv` を追加しなさい.
2. myshell にリダイレクト機能を追加しなさい.

```
#include <stdio.h> // perror() のため
#include <stdlib.h> // exit() のため
#include <string.h> // strcmp(), strchr() のため
#include <unistd.h> // fork(), exec(), close() のため
#include <sys/wait.h> // wait() のため
#include <ctype.h> // ispace() のため
#include <fcntl.h> // open() のため
#define MAXLINE 1000 // コマンド行の最大文字数
#define MAXARGS 60 // コマンド行文字列の最大数

int parse(char *p, char *args[]) { // コマンド行を解析する
    int i=0; // 解析後文字列の数
    for (;;) {
        while (isspace(*p)) // 空白が終わるまで進む
            *p++ = '\0'; // 前の文字列の終端に代用する
        if (*p=='\0' || i>=MAXARGS) break; // コマンド行の終端に到達で終了
        args[i++] = p; // 文字列を文字列配列に記録
        while (*p!='\0' && !isspace(*p)) // 文字列の最後まで進む
            p++;
    }
    args[i] = NULL; // 文字列配列の終端マーク
    return *p=='\0'; // 解析完了なら 1 を返す
}

void cdCom(char *path) { // cd コマンドを実行する
    if (path==NULL) { // 引数があるか調べて
        fprintf(stderr,"cd の引数が不足\n");
    } else if (chdir(path)<0) { // 親プロセスが chdir する
        perror(path);
    }
}

void setenvCom(char *args[]) { // setenv コマンドを実行する
    if (args[1]==NULL||args[2]==NULL) { // 引数があるか調べて
        fprintf(stderr,"setenv の引数が不足\n");
    } else if (setenv(args[1],args[2],1)<0) { // 環境変数を変更
        perror(args[1]);
    }
}

void unsetenvCom(char *name) { // unsetenv コマンドを実行する
    if (name==NULL) { // 引数があるか調べて
        fprintf(stderr,"unsetenv の引数が不足\n");
    } else if (unsetenv(name)<0) { // 環境変数を削除
        perror(name);
    }
}
```

```

}

char *ofile; // 入力リダイレクトファイル名
char *ifile; // 出力リダイレクトファイル名

void findRedirect(char *args[]) { // リダイレクトの指示を探す
    int i, j;
    ofile = ifile = NULL;
    for (i=j=0; args[i]!=NULL; i++) { // コマンド行の全文字列について
        if (strcmp(args[i], "<")==0) { // 入力リダイレクト発見
            ifile = args[++i]; // ファイル名を記録する
            if (ifile==NULL) break; // ファイル名が無かった
        } else if (strcmp(args[i], ">")==0) { // 出力リダイレクト発見
            ofile = args[++i]; // ファイル名を記録する
            if (ofile==NULL) break; // ファイル名が無かった
        } else { // どちらもでない
            args[j++] = args[i]; // 文字列を args に記録する
        }
    }
    args[j] = NULL;
}

void redirect(int fd, char *path, int mode) { // リダイレクト処理をする
    close(fd); // リダイレクトする fd をクローズ
    int nfd = open(path, mode, 0644); // 新しくオープン
    if (nfd<0) { // open がエラーなら
        perror(path); // エラーメッセージを表示
        exit(1); // 子プロセスを終了する
    }
    if (nfd!=fd) {
        fprintf(stderr, "リダイレクト処理でエラー\n");
        exit(1);
    }
}

void externalCom(char *args[]) { // 外部コマンドを実行する
    int pid, status;
    if ((pid = fork()) < 0) { // 新しいプロセスを作る
        perror("fork");
        exit(1); // fork 失敗
    }
    if (pid==0) { // 子プロセスなら
        if (ifile!=NULL) { // 入力リダイレクトがあれば
            redirect(0, ifile, O_RDONLY); // リダイレクトする
        }
        if (ofile!=NULL) { // 出力リダイレクトがあれば
            redirect(1, ofile, O_WRONLY|O_TRUNC|O_CREAT); // リダイレクトする
        }
        execvp(args[0], args); // コマンドを実行
        perror(args[0]);
        exit(1); // exec 失敗
    }
    while (wait(&status) != pid) // 親は子の終了を待つ

```

```
    ;
}

void execute(char *args[]) {
    // コマンドを実行する
    if (strcmp(args[0], "cd")==0) {
        // cd コマンド
        cdCom(args[1]);
    } else if (strcmp(args[0], "setenv")==0) {
        // setenv コマンド
        setenvCom(args);
    } else if (strcmp(args[0], "unsetenv")==0) {
        // unsetenv コマンド
        unsetenvCom(args[1]);
    } else {
        // 外部コマンド
        externalCom(args);
    }
}

int main() {
    char buf[MAXLINE+2];
    // コマンド行を格納する配列
    char *args[MAXARGS+1];
    // 解析結果を格納する文字列配列
    for (;;) {
        printf("Command: ");
        // プロンプトを表示する
        if (fgets(buf, MAXLINE+2, stdin) == NULL) {
            // コマンド行を入力する
            // EOF なら
            // 正常終了する
            break;
        }
        if (strchr(buf, '\n') == NULL) {
            // '\n' がバッファにない場合は
            fprintf(stderr, "行が長すぎる\n");
            // コマンド行が長すぎたので
            // 異常終了する
            return 1;
        }
        if (!parse(buf, args)) {
            // コマンド行を解析する
            // 文字列が多すぎる場合は
            // ループの先頭に戻る
            fprintf(stderr, "引数が多すぎる\n");
            continue;
        }
        findRedirect(args);
        // リダイレクトを見つける
        if (args[0] != NULL) {
            // コマンドがあることを確認して
            // コマンドを実行する
            execute(args);
        }
    }
    return 0;
}
```