

Slip 13

Program 1: Shell with `typeline` Command

```
#include <stdio.h>

#include <stdlib.h>

#include <string.h>

void typeline(char *option, char *filename) {

    FILE *file = fopen(filename, "r");

    if (file == NULL) {

        printf("File %s not found.\n", filename);

        return;

    }

    char line[1000];

    int n;

    if (strcmp(option, "-a") == 0) {

        while (fgets(line, sizeof(line), file)) {

            printf("%s", line);

        }

    } else if (option[0] == '+' && isdigit(option[1])) {

        n = atoi(option + 1);

        for (int i = 0; i < n; i++) {

            if (fgets(line, sizeof(line), file)) {

                printf("%s", line);

            }

        }

    }

    fclose(file);

}

int main() {

    char command[100], *args[10];

    while (1) {
```

```

printf("\nmyshell$ ");

fgets(command, 100, stdin);

command[strlen(command) - 1] = '\0'; // Remove newline


char *token = strtok(command, " ");

int i = 0;

while (token != NULL) {

    args[i++] = token;

    token = strtok(NULL, " ");

}

args[i] = NULL;


if (strcmp(args[0], "typeline") == 0) {

    typeline(args[1], args[2]);

} else if (strcmp(args[0], "exit") == 0) {

    exit(0);

} else {

    int pid = fork();

    if (pid == 0) {

        execvp(args[0], args);

        exit(0);

    } else {

        wait(NULL);

    }

}

return 0;

}

```

Program 2

```
#include <stdio.h>
```

```

struct process {

    int pid;

    int burst_time;

```

```

    int waiting_time;

    int turnaround_time;

};

void calculate_sjf(struct process p[], int n) {

    int total_waiting = 0, total_turnaround = 0;

    p[0].waiting_time = 0;

    for (int i = 1; i < n; i++) {

        p[i].waiting_time = p[i-1].waiting_time + p[i-1].burst_time;

    }

    for (int i = 0; i < n; i++) {

        p[i].turnaround_time = p[i].waiting_time + p[i].burst_time;

        total_waiting += p[i].waiting_time;

        total_turnaround += p[i].turnaround_time;

    }

    printf("\nPID\tBurst Time\tWaiting Time\tTurnaround Time\n");

    for (int i = 0; i < n; i++) {

        printf("%d\t%d\t\t%d\t\t%d\n", p[i].pid, p[i].burst_time, p[i].waiting_time, p[i].turnaround_time);

    }

    printf("\nAverage Waiting Time: %.2f", (float)total_waiting / n);

    printf("\nAverage Turnaround Time: %.2f", (float)total_turnaround / n);

}

void sort_by_burst_time(struct process p[], int n) {

    struct process temp;

    for (int i = 0; i < n-1; i++) {

        for (int j = i+1; j < n; j++) {

            if (p[i].burst_time > p[j].burst_time) {

                temp = p[i];

                p[i] = p[j];

                p[j] = temp;

            }

        }

    }

}

```

```
    }  
    }  
}  
}
```

```
int main() {  
    int n;  
    printf("Enter number of processes: ");  
    scanf("%d", &n);  
  
    struct process p[n];  
    for (int i = 0; i < n; i++) {  
        p[i].pid = i + 1;  
        printf("Enter burst time for process %d: ", p[i].pid);  
        scanf("%d", &p[i].burst_time);  
    }  
  
    sort_by_burst_time(p, n);  
    calculate_sjf(p, n);  
  
    return 0;  
}
```