

시스템 프로그래밍 실습 1-3 과제

이름 : 이준휘

학번 : 2018202046

교수 : 최상호 교수님

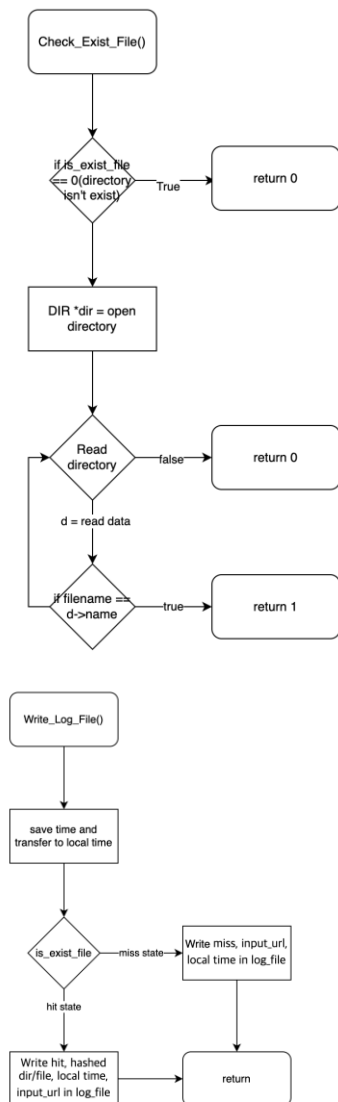
강의 시간 : 화

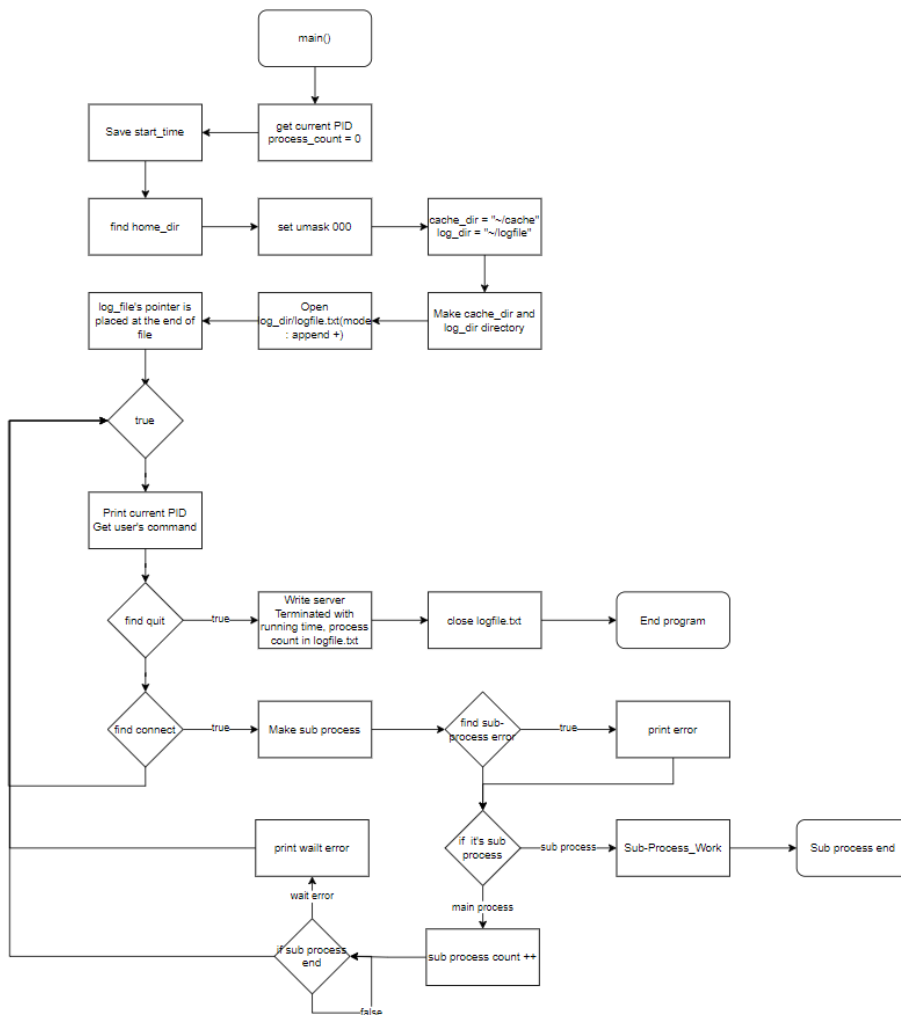
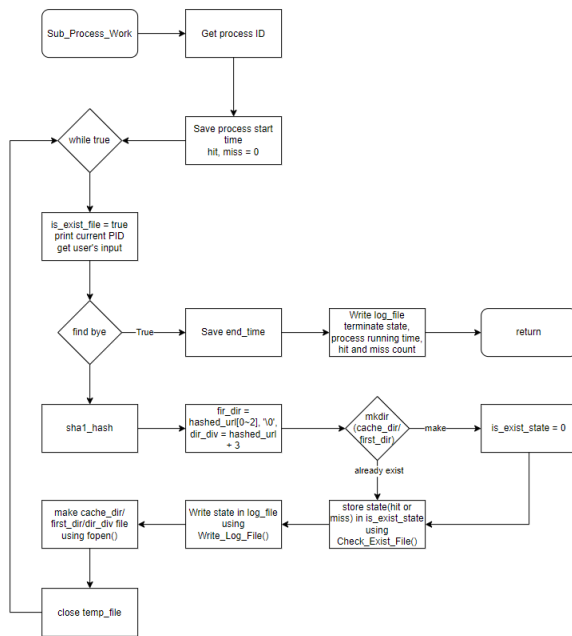
실습 분반 : 목 7, 8

1. Introduction

해당 과제는 1-2 과제에 덧붙여서 진행된다. 프로그램 실행 시 현재 command를 입력받는다. connect를 입력을 받을 시 sub-process를 생성하여 이전에 1-2에서 만든 URL 입력 동작을 수행한다. quit를 입력할 경우 메인 process는 종료된다. terminal 상에는 입력을 받을 때 현재 process ID가 출력되어야 하고, 출력한 프로세스 종료 시 logfile.txt에 해당 기록을 남긴다.

2. Flow Chart





3. Pseudo Code

```
Check_Exist_File(char *path, char *file_name, int is_exist_file){
```

```
    if(directory isn't exist)
```

```
        return 0;
```

```
    DIR *dir = Open path directory
```

```
    While(struct dirent *d = Read path directory){
```

```
        If(d->name == file name)
```

```
            Close directory and return 1;
```

```
    }
```

```
    Close directory and return 0;
```

```
}
```

```
Void Write_Log_File(File *log_file, char *input_url,
```

```
char *hashed_url_dir, char* hashed_url_file, int is_exist_file){
```

```
    time_t now;
```

```
    struct tm *ltp;
```

```
    ltp = current local time;
```

```
    if(miss state)
```

```
        Write miss, input_url, local time in log_file;
```

```
    Else
```

```
        Write hit, hashed dir/file, local time, input_url in log_file;
```

```
}
```

```

void Sub_Process_Work(char *input, char *char_dir, FILE *log_file){

    char[60] hashed_url;

    char[4] first_dir;

    char *dir_div;

    char[100] temp_dir;

    int is_exist_file, hit = 0, miss = 0;


    FILE *temp_file;

    pid_t current_pid = Current process ID;

    time_t start_process_time, end_process_time;

    Save start process time;

    while(true){

        print process id and get input;

        if(input is 'bye'){

            Save end process time;

            Write end – start process time, hit, miss count in logfile.txt;

            return;

        }

        is_exist_file = 1;

        hashed_url = hashed input using sha1;

        first_dir = { hashed_url[0~3], '\0' };

        dir_div = hashed_url + 3 address

        temp_dir = ~/cache/first_dir;

        if make temp_dir directory(permission = drwxrwxrwx)

            is_exist_file = 0;

        is_exist_file = hit or miss state;

```

```

    Write state, url, dir/file name, time in logfile.txt;

    temp_dir = ~/cache/first_dir/dir_div;

    temp_file = make temp_dir file and open file;

    temp_file close;

}

}

main(void){

    char[100] input, home_dir, cache_dir, log_dir, temp_dir;

    int sub_process_count = 0;

    time_t start_time, end_time;

    FILE *log_file;

    pid_t pid, current_pid = Current Process ID;

    Save start time;


    home_dir = ~;

    cache_dir = ~/cache;

    log_dir = ~/logfile;

    set umask 000;

    make cache and log directory(permission = drwxrwxrwx);

    temp_dir = ~/logfile/logfile/txt;

    make log file(mode : a+);

    log_file's pointer is placed in the end of file;


    while(true){

        print current pid and Get command;

```

```

if(command == quit){

    Save end time;

    Write server terminated with running time, sub process count in log_file;

    end program;

}

if(command = connect){

    pid = make sub process;

    if(failed to make sub process)

        print error;

    if(Current process is Sub process)

        Do Sub_Process_Work and end process;

    if(Current process is main process)

        Process count++ and wait until Sub_Process end;

}

```

4. 결과 화면

```

//Sub_Process_Work
//Input : char *input -> get URL from user,
//char *cache_dir -> .cache path
//File log_file -> write ./logfile.logfile.txt,
//Output : void
//Purpose : make hashed url directory and file in ./cache/
//Write hit and miss state in ./logfile.logfile.txt
//=====
void Sub_Process_Work(char *input, char *cache_dir, FILE *log_file)
{
    char hashed_url[100]; //store hashed URL string
    char first_dir[50]; //directory that will be made in cache directory
    char *dir_div; //separate point of hashed URL name
    char temp_dir[100]; //path to used when it makes directory or file

    int is_exist_file; //State directory/file is exist
    int hit = 0, miss = 0; //Count hit and miss

    FILE *temp_file;
    printf("current pid = %d\n", getpid()); //Current process id
    time_t start_process_time, end_process_time; //Process start and end time
    time(&start_process_time); //Check process start time

    while(1)
    {
        printf("%d\n", input); //Print process id and get URL
        scanf("%s", input); //If it's 'bye' command, write log at logfile.txt and end function
        if(strlen(input) == 0)
        {
            time(&end_process_time); //Check end process time
            printf("log_file : [terminated] run time : %d\n", (int)(end_process_time - start_process_time), hit, miss); //Write process execution time, hit, miss in logfile.txt
            return; //End program
        }

        is_exist_file = 0;
        if(strlen(input) > 0)
        {
            hashed_url = input; //URL -> hashed URL
            dir_div = hashed_url; //Directory name <- hashed url[-2]
            first_dir[0] = '\0'; //File name pointer
            strcpy(temp_dir, cache_dir); //Make directory ./cache/Directory name (permission : rwxrwxrwx)
            strcat(temp_dir, dir_div);
            if(fopen(temp_dir, "w") == 0) //Directory isn't already exist, is_exist_file is 0
            {
                is_exist_file = 1;
                printf("is_exist_file : %d\n", is_exist_file); //Check ./cache/Directory name/file name is exist
                printf("log_file : [terminated] run time : %d\n", (int)(end_process_time - start_process_time), hit, miss); //Write the state(hit or miss) in logfile.txt
                utime(temp_dir, NULL); //Make empty file ./cache/Directory name/File name (permission : rwxrwxrwx)
                temp_file = fopen(temp_dir, "w");
                fclose(temp_file);
            }
        }
    }

    void main(int argc, char *argv[])
    {
        //Store input Data(URL or bye)
    }
}

```

해당 사진은 Sub_Process_Work를 찍은 것이다. 해당 함수는 기존의 1-2에서 반복되는 부분을 함수로 바꾸고 현재의 PID를 추가적으로 출력하는 것으로 바뀌었으며,

또한 해당 함수가 종료할 경우 함수의 시간을 기록함으로써 sub process가 얼마동안 동작하였는지 logfile.txt에 기록한다.

```

1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100
101
102
103
104
105
106
107
108
109
110
111
112
113
114
115
116
117
118
119
120
121
122
123
124
125
126
127
128
129
130
131
132
133
134
135
136
137
138
139
140
141
142
143
144
145
146
147
148
149
150
151
152
153
154
155
156
157
158
159
160
161
162
163
164
165
166
167
168
169
170
171
172
173
174
175
176
177
178
179
180
181
182
183
184
185
186
187
188
189
190
191
192
193
194
195
196
197
198
199
200
201
202
203
204
205
206
207
208
209
210
211
212
213
214
215
216
217
218
219
220
221
222
223
224
225
226
227
228
229
230
231
232
233
234
235
236
237
238
239
240
241
242
243
244
245
246
247
248
249
250
251
252
253
254
255
256
257
258
259
260
261
262
263
264
265
266
267
268
269
270
271
272
273
274
275
276
277
278
279
280
281
282
283
284
285
286
287
288
289
290
291
292
293
294
295
296
297
298
299
300
301
302
303
304
305
306
307
308
309
310
311
312
313
314
315
316
317
318
319
320
321
322
323
324
325
326
327
328
329
330
331
332
333
334
335
336
337
338
339
340
341
342
343
344
345
346
347
348
349
350
351
352
353
354
355
356
357
358
359
360
361
362
363
364
365
366
367
368
369
370
371
372
373
374
375
376
377
378
379
380
381
382
383
384
385
386
387
388
389
390
391
392
393
394
395
396
397
398
399
400
401
402
403
404
405
406
407
408
409
410
411
412
413
414
415
416
417
418
419
420
421
422
423
424
425
426
427
428
429
430
431
432
433
434
435
436
437
438
439
440
441
442
443
444
445
446
447
448
449
450
451
452
453
454
455
456
457
458
459
460
461
462
463
464
465
466
467
468
469
470
471
472
473
474
475
476
477
478
479
480
481
482
483
484
485
486
487
488
489
490
491
492
493
494
495
496
497
498
499
500
501
502
503
504
505
506
507
508
509
510
511
512
513
514
515
516
517
518
519
520
521
522
523
524
525
526
527
528
529
530
531
532
533
534
535
536
537
538
539
540
541
542
543
544
545
546
547
548
549
550
551
552
553
554
555
556
557
558
559
560
561
562
563
564
565
566
567
568
569
570
571
572
573
574
575
576
577
578
579
580
581
582
583
584
585
586
587
588
589
590
591
592
593
594
595
596
597
598
599
600
601
602
603
604
605
606
607
608
609
610
611
612
613
614
615
616
617
618
619
620
621
622
623
624
625
626
627
628
629
630
631
632
633
634
635
636
637
638
639
640
641
642
643
644
645
646
647
648
649
650
651
652
653
654
655
656
657
658
659
660
661
662
663
664
665
666
667
668
669
670
671
672
673
674
675
676
677
678
679
680
681
682
683
684
685
686
687
688
689
690
691
692
693
694
695
696
697
698
699
700
701
702
703
704
705
706
707
708
709
710
711
712
713
714
715
716
717
718
719
720
721
722
723
724
725
726
727
728
729
730
731
732
733
734
735
736
737
738
739
740
741
742
743
744
745
746
747
748
749
750
751
752
753
754
755
756
757
758
759
760
761
762
763
764
765
766
767
768
769
770
771
772
773
774
775
776
777
778
779
780
781
782
783
784
785
786
787
788
789
790
791
792
793
794
795
796
797
798
799
800
801
802
803
804
805
806
807
808
809
810
811
812
813
814
815
816
817
818
819
820
821
822
823
824
825
826
827
828
829
830
831
832
833
834
835
836
837
838
839
840
841
842
843
844
845
846
847
848
849
850
851
852
853
854
855
856
857
858
859
860
861
862
863
864
865
866
867
868
869
870
871
872
873
874
875
876
877
878
879
880
881
882
883
884
885
886
887
888
889
890
891
892
893
894
895
896
897
898
899
900
901
902
903
904
905
906
907
908
909
910
911
912
913
914
915
916
917
918
919
920
921
922
923
924
925
926
927
928
929
930
931
932
933
934
935
936
937
938
939
940
941
942
943
944
945
946
947
948
949
950
951
952
953
954
955
956
957
958
959
960
961
962
963
964
965
966
967
968
969
970
971
972
973
974
975
976
977
978
979
980
981
982
983
984
985
986
987
988
989
990
991
992
993
994
995
996
997
998
999
1000

```

위의 사진은 메인 함수를 표시한 것이다. 해당 함수는 logfile.txt를 여는 동작까지는 기존과 동일하지만 이후의 동작은 달라진다. 이후에 while문을 통해 반복적으로 현재의 PID를 출력하고 입력을 받는다. 만약 입력이 quit일 경우에는 종료 시간을 측정, logfile.txt에 Server가 terminate됨을 알리고 동작 시간, sub process 작동 횟수를 기록한다. 만약 connect를 입력을 받은 경우 fork() 함수를 통해 sub process를 만들고 sub process는 1-2의 작업에 해당하는 Sub_Process_Work() 함수를 수행한 뒤 마친다. main process는 process count를 1 증가시키고, sub process가 끝날 때까지 기다린다.

```

1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100
101
102
103
104
105
106
107
108
109
110
111
112
113
114
115
116
117
118
119
120
121
122
123
124
125
126
127
128
129
130
131
132
133
134
135
136
137
138
139
140
141
142
143
144
145
146
147
148
149
150
151
152
153
154
155
156
157
158
159
160
161
162
163
164
165
166
167
168
169
170
171
172
173
174
175
176
177
178
179
180
181
182
183
184
185
186
187
188
189
190
191
192
193
194
195
196
197
198
199
200
201
202
203
204
205
206
207
208
209
210
211
212
213
214
215
216
217
218
219
220
221
222
223
224
225
226
227
228
229
230
231
232
233
234
235
236
237
238
239
240
241
242
243
244
245
246
247
248
249
250
251
252
253
254
255
256
257
258
259
260
261
262
263
264
265
266
267
268
269
270
271
272
273
274
275
276
277
278
279
280
281
282
283
284
285
286
287
288
289
290
291
292
293
294
295
296
297
298
299
300
301
302
303
304
305
306
307
308
309
310
311
312
313
314
315
316
317
318
319
320
321
322
323
324
325
326
327
328
329
330
331
332
333
334
335
336
337
338
339
340
341
342
343
344
345
346
347
348
349
350
351
352
353
354
355
356
357
358
359
360
361
362
363
364
365
366
367
368
369
370
371
372
373
374
375
376
377
378
379
380
381
382
383
384
385
386
387
388
389
390
391
392
393
394
395
396
397
398
399
400
401
402
403
404
405
406
407
408
409
410
411
412
413
414
415
416
417
418
419
420
421
422
423
424
425
426
427
428
429
430
431
432
433
434
435
436
437
438
439
440
441
442
443
444
445
446
447
448
449
450
451
452
453
454
455
456
457
458
459
460
461
462
463
464
465
466
467
468
469
470
471
472
473
474
475
476
477
478
479
480
481
482
483
484
485
486
487
488
489
490
491
492
493
494
495
496
497
498
499
500
501
502
503
504
505
506
507
508
509
510
511
512
513
514
515
516
517
518
519
520
521
522
523
524
525
526
527
528
529
530
531
532
533
534
535
536
537
538
539
540
541
542
543
544
545
546
547
548
549
550
551
552
553
554
555
556
557
558
559
560
561
562
563
564
565
566
567
568
569
570
571
572
573
574
575
576
577
578
579
580
581
582
583
584
585
586
587
588
589
590
591
592
593
594
595
596
597
598
599
600
601
602
603
604
605
606
607
608
609
610
611
612
613
614
615
616
617
618
619
620
621
622
623
624
625
626
627
628
629
630
631
632
633
634
635
636
637
638
639
640
641
642
643
644
645
646
647
648
649
650
651
652
653
654
655
656
657
658
659
660
661
662
663
664
665
666
667
668
669
670
671
672
673
674
675
676
677
678
679
680
681
682
683
684
685
686
687
688
689
690
691
692
693
694
695
696
697
698
699
700
701
702
703
704
705
706
707
708
709
710
711
712
713
714
715
716
717
718
719
720
721
722
723
724
725
726
727
728
729
730
731
732
733
734
735
736
737
738
739
740
741
742
743
744
745
746
747
748
749
750
751
752
753
754
755
756
757
758
759
760
761
762
763
764
765
766
767
768
769
770
771
772
773
774
775
776
777
778
779
780
781
782
783
784
785
786
787
788
789
790
791
792
793
794
795
796
797
798
799
800
801
802
803
804
805
806
807
808
809
810
811
812
813
814
815
816
817
818
819
820
821
822
823
824
825
826
827
828
829
830
831
832
833
834
835
836
837
838
839
840
841
842
843
844
845
846
847
848
849
850
851
852
853
854
855
856
857
858
859
860
861
862
863
864
865
866
867
868
869
870
871
872
873
874
875
876
877
878
879
880
881
882
883
884
885
886
887
888
889
890
891
892
893
894
895
896
897
898
899
900
901
902
903
904
905
906
907
908
909
910
911
912
913
914
915
916
917
918
919
920
921
922
923
924
925
926
927
928
929
930
931
932
933
934
935
936
937
938
939
940
941
942
943
944
945
946
947
948
949
950
951
952
953
954
955
956
957
958
959
960
961
962
963
964
965
966
967
968
969
970
971
972
973
974
975
976
977
978
979
980
981
982
983
984
985
986
987
988
989
990
991
992
993
994
995
996
997
998
999
1000

```

위의 사진은 해당 프로그램을 동작시킨 사진이다. 기존에 cache와 logfile이 없는 상태에서 해당 함수를 수행하였다. 처음 동작하였을 때 현재 프로세스와 CMD를

입력 받는 동작이 제대로 수행되었고 connect를 입력 받은 경우 Sub process를 발생시키기 때문에 pid가 다르게 출력된다. URL 2개를 입력받고 해당 프로세스를 종료시키면 다시 원래 pid가 출력된다. 입력의 경우 첫 번째 프로세스에서는 miss 2개, 두 번째 프로세스에서는 hit 1개, miss 1개를 입력하였다. logfile.txt를 살펴본 결과 2개의 프로세스가 위의 언급한 상태 그대로 기록되었고 Server를 종료할 경우 해당되는 상태 또한 제대로 기록된 것을 볼 수 있다. 이를 통해 해당 프로그램이 목적에 맞게 구현된 것을 확인하였다.

5. 고찰

해당 과제를 통해서 getpid()함수를 통해 현재의 프로세스에 대한 정보를 알 수 있다는 사실을 알게 되었다. 또한 fork()함수를 사용할 경우 2개의 return값을 통해 각각의 프로세스가 동작한다는 메커니즘을 이해할 수 있었다. wait() 함수나 waitpid()를 직접 사용해 보면서 어떻게 하면 child process가 끝날 때까지 main process가 동작하지 않을지에 대해 고민해보았으며, child process가 main process의 자원을 복사하여 사용한다는 사실 또한 알게 되었다.

6. Reference

강의 자료만을 참고