Homework #5 **M1522.000800 System Programming**

Sample Solution

Question 1

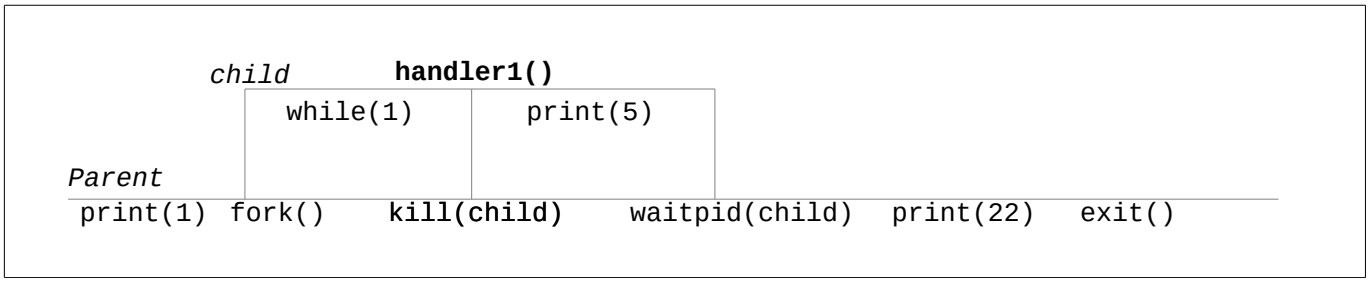
Signal Handling Issues

What is one possible output of the following program? Show the flow-chart of this program, too.

```
pid_d pid;
int counter = 1;

void handler1(int sig) {
    counter = 5;
    printf(“%d”, counter);
    fflush(stdout);
    exit(0);
}

int main() {
    signal(SIGUSR1, handler1);

    printf(“%d”, counter);
    fflush(stdout);

    if ((pid = fork()) == 0) {
        while(1) {};
    }
    kill(pid, SIGUSR1);
    waitpid(-1, NULL, 0);
    counter = counter + 1;
    printf(“%d%d”, counter, counter);
    exit(0);
}
```

The output is 1522.

< Flow Chart >



Question 2

Classes of exceptions

There are four classes of exceptions, with different causes, return behavior and synchronicity. Fill in the missing information in the following table.

Class	Cause	Sync/Async	Return to
Abort	Nonrecoverable error (e.g., memory parity errors)	sync	never return
trap	Intentional exceptions (e.g., system call)	sync	return to next instruction
fault	Potentially recoverable error (e.g., page fault)	sync	might return to current instruction
interrupt	Interrupt signal from I/O device (e.g., disk, keyboard, mouse,...)	async	Return to next instruction

Question 3

Exception handling

User programs run in user mode. Typically, devices such as the disk cannot be accessed directly in user mode. A user program invokes a system call to have the operating system perform the low-level work.

(a) Which exception mechanism is used for system calls?

=> Traps are used for system calls.

(b) On Linux/IA32 systems, which instruction is used to signal a system call?

=> The instruction to signal a system call is ***`int 0x80`***.

(c) How are parameters passed when invoking a system call?

=> Parameters are passed through registers. By convention, %eax contains the system call number, and registers %ebx, %ecx, %edx, %esi, %edi, and %ebp contain up to 6 arbitrary arguments.

Question 4

Process Scheduling Algorithms

The following is a table which informs a few things for each processes.

Process	Arrival Time	Burst Time	Priority
P1	0	6	2
P2	2	4	1
P3	5	5	3
P4	6	3	4

Then, draw the *Gantt Chart* and calculate an *average waiting time, average turn-around time, throughput* for each scheduling algorithms.

(a) **First-Come, First-Served (FCFS)**

P1	P2	P3	P4

0 6 10 15

=> average waiting time : start time to execute – arrival time
((0 – 0) + (6 – 2) + (10 – 5) + (15 – 6)) / 4 = (4 + 5 + 9) / 4 = 4.5
=> average turn-around time : complete time to execute – arrival time
((6 – 0) + (10 – 2) + (15 – 5) + (18 – 6)) / 4 = (6 + 8 + 10 + 12) / 4 = 9
=> throughput : 4 processes / 18 time quantum

(b) **Shortest-Job-First (SJF)**

P1	P4	P2	P3

0 6 9 13

=> average waiting time : start time to execute – arrival time
((0 – 0) + (9 – 2) + (13 – 5) + (6 – 6)) / 4 = (7 + 8) / 4 = 3.75
=> average turn-around time : complete time to execute – arrival time
((6 – 0) + (13 – 2) + (18 – 5) + (9 – 6)) / 4 = (6 + 11 + 13 + 3) / 4 = 8.25
=> throughput : 4 processes / 18 time quantum

(c) **Shortest-Remaining-Time-First (SRTF) : assume that preemptive &prefer not to switch**

P1	P1	P4	P2	P4

0 2 6 9 13

=> average waiting time : start time to execute – arrival time
((0 – 0) + (9 – 2) + (13 – 5) + (6 – 6)) / 4 = (7 + 8) / 4 = 3.75
=> average turn-around time : complete time to execute – arrival time
((6 – 0) + (13 – 2) + (18 – 5) + (9 – 6)) / 4 = (6 + 11 + 13 + 3) / 4 = 8.25
=> throughput : 4 processes / 18 time quantum

(d) **Priority Scheduling : assume that preemptive**

P1	P2	P1	P3	P4
0	2	6	10	15

=> average waiting time : start time to execute – arrival time (start time to wait)
((6 – 2) + (2 – 2) + (10 – 5) + (15 – 6)) / 4 = (4 + 5 + 9) / 4 = 4.5
=> average turn-around time : complete time to execute – arrival time
((10 – 0) + (6 – 2) + (15 – 5) + (18 – 6)) / 4 = (10 + 4 + 10 + 12) / 4 = 9
=> throughput : 4 processes / 18 time quantum

(e) **Round Robin (RR) with Time Quantum = 3**

P1	P2	P3	P4	P1	P2	P3
0	3	6	9	12	15	16

=> average waiting time : start time to execute – arrival time (start time to wait)
((12 – 3) + (15 – 6 + 3 - 2) + (16 – 9 + 6 - 5) + (9 – 6)) / 4 = (9 + 10 + 8 + 3) / 4 = 7.5
=> average turn-around time : complete time to execute – arrival time
((15 – 0) + (16 – 2) + (18 – 5) + (12 – 6)) / 4 = (15 + 14 + 13 + 6) / 4 = 12
=> throughput : 4 processes / 18 time quantum