Name: Niranjan Malla
Class: CS470
Lab: #3

Consider the following piece of C code:

```
void main( ) {
    fork( );
    fork( );
    exit( );
}
```

1. How many child processes are created upon execution of this program?

Answer:

There are 3 child processes created upon the execution of the given program.

[(2^n -1)] helps to find the number of child process created where n is the number of fork in the given program.

When you run the top command in the terminal, it will display real-time information about running processes and system usage, such as CPU usage, memory usage, and process information.

2. When you start a browser, you will notice the browser process appear in the top display. What does it consume?

Ans: It gives live information about CPU and Memory.

Processes: 465 total, 3 running, 462 sleeping, 2234 threads 18:32:35
Load Avg: 1.84, 1.53, 1.51 CPU usage: 8.82% user, 5.22% sys, 85.94% idle SharedLibs: 426M resident, 87M data, 28M linkedit. MemRegions: 197304 total, 1714M resident, 146M private, 1254M shared.
PhysMem: 7430M used (1232M wired, 2448M compressor), 200M unused. VM: 188T vsize, 4184M framework vsize, 522405(0) swapins, 634514(0) swapouts. Networks: packets: 4764011/3225M in, 2409723/543M out.
Disks: 1307375905/22T read, 3405067/54G written.

PID	COMMAND	%CPU	TIME	#TH	#WQ	#PORT	MEM	PURG	CMPRS	PGRP	PPID	STATE	BOOSTS	%CPU_ME	%CPU_OTHRS	UID	FAULTS	COW	MSGSENT	MSGRECV	SYSBSD	SYSMACH
12700	zoom.us	32.1	48:24.98	48/1	5	1609	272M	64K	130M	12700	1	running	*0[32864+]	1.30538	0.03524	501	4943145+	761	8968328+	3190561+	30488454+	15277070+
204	coreaudiod	14.0	35:10.61	11	3	1425	50M	0B	35M	204	1	sleeping	*0[1]	0.00000	1.06415	202	1067251	200	17782822+	13481301+	69691162+	45250055+
156	WindowServer	12.3	05:22:13	23	6	3349+	954M-	11M	223M	156	1	sleeping	*0[1]	0.18537	0.41511	88	11791144+	33385	75010676+	1197991490+	2147483647	2147483647
0	kernel_task	7.2	02:40:02	514/8	0	0	3216K+	0B	0B	0	0	running	0[0]	0.00000	0.00000	0	10898	0	149208081+	88781306+	0	0
13895	top	6.4	02:34.81	1/1	0	46-	7649K	0B	1168K	13895	13882	running	*0[1]	0.00000	0.00000	0	1092316+	70	29024170+	14512077+	4414607+	22989791+
162	runningboard	4.1	02:01.10	8	7	574-	7889K	0B	1152K	162	1	sleeping	*6-[1]	0.04491	3.12394	0	119057	89	303343+	280735+	4629487+	662110+
10195	contactsd	2.5	01:45.69	5	4	421-	10M	1536K	3296K	10195	1	sleeping	*0[36221+]	0.00000	2.16549	501	155627	99	212141+	193326+	3310219+	1016328+
11112	Terminal	2.1	03:20.95	11	5	436+	70M+	21M	23M-	11112	1	sleeping	*0[1195+]	0.02702	0.03962	501	1037637+	365	1055880+	179633+	1415518+	3305327+
14085	mdworker_sha	1.9	00:00.11	4	1	53+	2449K+	0B	0B	14085	1	sleeping	*0[1]	1.29734	0.00000	501	1805+	91+	511+	211+	1113+	583+
13842	Google Chrom	1.5	00:45.31	23	1	250	101M	0B	72M	3802	3802	sleeping	*0[4]	0.00000	0.00000	501	39291	684	552258+	175213+	1603967+	1367750+
13081	Google Chrom	1.5	02:19.02	22	1	247	101M	0B	73M	3802	3802	sleeping	*0[4]	0.00000	0.00000	501	68157	773	2303157+	728141+	6466969+	5683609+
13474	Google Chrom	1.5	01:23.71	22	1	199	53M	0B	41M	3802	3802	sleeping	*0[3]	0.00000	0.00000	501	27050	668	1251181+	375432+	3750805+	3076557+
660	mds_stores	1.4	10:50.53	6	4	125	42M+	0B	28M	660	1	sleeping	*0[1]	0.00000	1.17980	0	1581023+	381	282881+	263964+	9562146+	781531+
705	cfprefsd	1.1	00:47.72	3	2	526-	3969K	32K+	1920K-	705	1	sleeping	*30[724]	0.00000	0.18779	501	79267+	65	184420+	188046+	2068760+	409972+
4301	Microsoft Wo	1.0	06:09.46	23	7	1502	534M	24M	278M	4301	1	sleeping	*0[3489]	0.00000	0.00000	501	2139168	3928	2810193+	782839+	2697932+	6078628+
6670	tccd	0.9	00:32.42	3	2	71	5825K	0B	3184K	6670	1	sleeping	*0[16204+]	0.00000	0.84096	501	72172	85	94384+	96199+	1961627+	256754+
115	mds	0.8	05:35.90	9	6	476	43M-	0B	29M	115	1	sleeping	*0[1]	0.00000	0.16637	0	1441609+	547	449199+	330933+	13341252+	1090150+
11351	bluetoothd	0.7	03:24.02	9	3	336	15M	176K	7296K	11351	1	sleeping	*0[1]	0.36870	0.02112	0	29839	148	307706+	351514+	8291960+	1194848+
95	fseventsd	0.5	02:51.20	14	1	184	6129K	0B	1920K	95	1	sleeping	*0[1]	0.00000	0.00000	0	443600+	65	736616+	276878+	7033832+	1295017+
3814	Google Chrom	0.5	23:08.90	24	5	299	487M	0B	174M	3802	3802	sleeping	*1[2]	0.00000	0.00000	501	2657571	787	17621943	28324065+	27224144+	68757144+
245	TouchBarServ	0.5	43:00.93	4	2	481+	28M+	2304K-	11M-	245	1	sleeping	*0[1]	0.12621	0.00847	0	173190+	162	1901581+	2540044+	247202093+	248546618+
14084	mdworker_sha	0.5	00:00.09	3	1	51	2353K	0B	0B	14084	1	sleeping	*0[1]	0.02004	0.00000	501	1550+	89	476+	199+	705+	530+
101	powerd	0.5	00:56.95	4	3	140+	4625K+	0B	1232K-	101	1	sleeping	*0[1]	0.00000	0.35635	0	330203+	118	721087+	507987+	1733419+	1442982+
4214	Code Helper	0.4	06:48.98	12	4	168	106M	0B	57M	4211	4211	sleeping	*1[5]	0.00000	0.00000	501	362886	548	2361016	21389942+	16303751+	38741862+
14083	mdworker_sha	0.4	00:00.07	3	1	51	2321K	0B	0B	14083	1	sleeping	*0[1]	0.02879	0.00000	501	1378+	88	448+	191+	599+	479+
421	locationd	0.3	02:13.86	6	3	345	10M	256K	3888K	421	1	sleeping	*0[74404+]	0.00000	0.28271	205	199640	156	212022	317163+	4258151+	981280+
90	logd	0.3	18:25.49	4	3	1521-	25M	0B	19M-	90	1	sleeping	*0[1]	0.00000	0.00000	0	518560+	69	30493474+	32407732+	68768868+	30649340+
3816	Google Chrom	0.3	05:29.31	12	1	114	36M-	0B	15M	3802	3802	sleeping	*0[3]	0.00000	0.00000	501	673995	690	4748140	3452208	13112141+	8200830+
1	launchd	0.2	04:58.65	3	2	3162-	23M	0B	8368K	1	0	sleeping	0[0]	0.00000	0.13724	0	395275	8929	855779+	857691+	5025609+	2689248+
3802	Google Chrom	0.2	30:33.36	39	2	1472	284M	0B	136M	3802	1	sleeping	*0[5142]	0.00000	0.00000	501	5407806	69968	34653667	15989408	44308528	105965496+
10965	com.apple.ap	0.2	00:17.90	6	2	431	51M	0B	32M	10965	1	sleeping	*4287[5]	0.00000	0.00000	501	131208+	287	300960	78554+	388160+	496432+
1480	Finder	0.2	04:50.71	6	3	722	100M	0B	59M	1480	1	sleeping	*0[5830]	0.00000	0.00000	501	1675202+	337	6219212+	1379337+	4200936+	13600145+
11986	PerfPowerSer	0.1	01:04.72	5	3	377+	16M+	256K	3552K	11986	1	sleeping	0[2227]	0.00000	0.16850	0	42659+	176	1172582	756249+	1034520+	1779658+
11001	com.apple.ap	0.1	00:23.97	4	2	443	46M	0B	17M	11001	1	sleeping	*7324[889]	0.00000	0.00000	501	118796+	762	632169	111372+	560115+	863047+
713	qemu-system-	0.1	03:41.95	7	1	33	1952M	0B	1894M	439	439	sleeping	*0[1]	0.00000	0.00000	0	3985231+	301	142	37	8335659+	4745
10169	Siri	0.1	00:05.33	3	1	163	9394K	0B	4480K-	10169	1	sleeping	*0[1354+]	0.00000	0.00000	501	26342+	144	84217+	13583+	49259+	185680+
1494	multipass.gu	0.1	01:11.47	13	1	214	55M+	0B	44M	1494	1	sleeping	*0[1]	0.00000	0.00000	501	140299+	288	302957+	178020+	2438275+	2397788+
193	airportd	0.1	08:12.41	10	8	904+	15M+	0B	6592K	193	1	sleeping	*2449[2]	0.02315	0.00000	0	685484+	270	2751409+	1540416+	6799877+	4001003+
450	audioclocksy	0.1	00:43.17	5	4	69	4833K	0B	2544K	450	1	sleeping	*0[1]	0.04307	0.00000	0	18837	65	324957+	717655+	1063455+	1602843+
813	sharingd	0.1	01:51.03	5	1	361	19M	0B	9504K	813	1	sleeping	*0[1]	0.00000	0.04644	501	170816	277	911941+	428507+	2528080+	3123507+
11990	PerfPowerSer	0.1	00:00.30	2	1	75	2273K+	0B	1152K-	11990	1	sleeping	0[118]	0.00000	0.08227	0	2737+	79	1514+	1298+	4682+	3165+
14086	AddressBookS	0.0	00:00.23	4	3	100-	6049K-	448K	0B	14086	1	sleeping	0[3]	0.02136	0.00000	501	1552	128	310+	79+	3048+	921+
128	launchservic	0.0	00:56.54	3	2	539+	7905K+	0B	1648K-	128	1	sleeping	*11[665370]	0.00000	0.00000	0	119877+	82	1026922+	863541+	2664817+	1661595+
9233	com.apple.hi	0.0	00:02.87	2	1	72	2385K	0B	912K	9233	1	sleeping	*0[11859+]	0.00000	0.00000	501	11114	69	25727+	24418+	125861+	39303+
3817	Google Chrom	0.0	00:10.08	8	1	88	19M	0B	15M	3802	3802	sleeping	*0[3]	0.00000	0.00000	501	85823	1851	160567	74970	235233+	327756+
439	multipassd	0.0	00:43.09	10	1	69	28M	0B	25M	439	1	sleeping	*0[1]	0.00000	0.00000	0	50712	297	458	603	2001155+	2159
811	ControlCente	0.0	02:12.41	5	1	611	50M	0B	26M	811	1	sleeping	*0[110087+]	0.00000	0.03516	501	595243+	1443	1971739+	826241+	1750256+	2982789+
4230	Code Helper	0.0	00:25.70	22	1	174	52M	0B	41M	4211	4211	sleeping	*0[1]	0.00000	0.00000	501	379899	502	62628+	26975+	619257+	631436+
137	dasd	0.0	00:45.36	3	2	97-	9169K-	0B	2480K	137	1	sleeping	0-[2375]	0.00000	0.04048	0	133574	127	42500+	27501	834031+	100947+
810	nearbyd	0.0	00:32.79	7	5	95	4513K	0B	2240K	810	1	sleeping	*9[6379]	0.00000	0.03955	268	22847	112	9687	102952+	1153763+	113576+
154	cfprefsd	0.0	00:35.92	4	3	740-	3697K-	0B-	1136K	154	1	sleeping	*1[1460]	0.00000	0.00000	0	94701+	63	87766	101311+	1436321+	255017+
149	notifyd	0.0	00:37.52	2	1	768-	3377K	0B	992K	149	1	sleeping	*0[1]	0.00000	0.00000	0	18736	46	654980	782946+	2397564+	1483792+
779	accountsd	0.0	01:19.97	5	4	163-	16M	256K	8656K	779	1	sleeping	0[52961]	0.00000	0.00000	501	376807	293	127844	92332+	1613616+	840113+
10252	routined	0.0	01:17.08	5	4	214-	17M-	1536K	21M	10252	1	sleeping	0[121]	0.00000	0.00000	501	86103	238	27080+	12667+	1335825+	805277+
6710	ViewBridgeAu	0.0	00:01.92	2	1	151	5025K	0B	2448K	6710	1	sleeping	*0[16440+]	0.00000	0.00000	501	27344	101	53146+	20872+	47391+	137215+
4234	Code Helper	0.0	00:09.67	14	1	59	21M	0B	17M	4211	4230	sleeping	*0[1]	0.00000	0.00000	501	65225	580	94	98	146374+	4675
10767	amsengagemen	0.0	00:03.60	3	2	136	13M+	0B	11M	10767	1	sleeping	0[92]	0.00000	0.00000	501	20445+	260	6549+	2169	107238+	25097+

3. How much memory is available in the system?

Ans:

200 M unused is available in the system.

4. Which process consumes the most CPU?

Ans: Zoom.us (32.1%)

5. Which process has the most memory?

Ans: zoom.us has the most memory at a given run time.

6. Could you please explain the following commands?

apt-get, yum, wget, gzip, tar, rar

Ans:

1. **apt-get**: This is a package manager for Debian-based systems, such as Ubuntu, used to install, remove and update software packages.
2. **yum**: This is a package manager for Red Hat-based systems, such as Fedora, used to install, remove and update software packages.
3. **wget**: This is a command line utility used to download files from the internet.
4. **gzip**: This is a file compression utility that reduces the size of a file using the gzip algorithm.
5. **tar**: This is a utility used to combine multiple files into a single archive file, often with the extension .tar.
6. **rar**: This is a file compression utility used to compress and archive files into a single .rar file.

7. Write a program that will generate a child process. In a loop, the child process writes "I am a child process" 200 times and the parent process repeatedly prints "I am a parent process" in a loop.

```
ubuntu@primary:~$ cat lab3_7.c
#include <stdio.h>
#include <unistd.h>
#include <sys/types.h>

int main(void) {
    pid_t pid = fork();
    int i;

    if (pid == 0) {
        // Child process
        for (i = 0; i < 200; i++) {
            printf("I am a child process\n");
        }
    } else {
        // Parent process
        for (i = 0; i < 200; i++) {
            printf("I am a parent process\n");
        }
    }

    return 0;
}
```

```
**** System restart required ***
Last login: Sat Jan 28 23:21:39 2023 from 192.168.64.1
[ubuntu@primary:~$ touch lab3_7.c
[ubuntu@primary:~$ nano lab3_7.c
[ubuntu@primary:~$ gcc lab3_7.c
[ubuntu@primary:~$ ./a.out
I am a parent process
I am a parent process
I am a parent process
I am a parent process
I am a parent process
I am a parent process
I am a parent process
I am a parent process
I am a parent process
I am a parent process
I am a parent process
I am a parent process
I am a parent process
I am a parent process
I am a parent process
I am a parent process
I am a parent process
I am a parent process
I am a parent process
I am a parent process
I am a parent process
I am a parent process
I am a parent process
I am a parent process
I am a parent process
I am a parent process
I am a parent process
I am a parent process
I am a parent process
I am a parent process
I am a parent process
I am a parent process
I am a parent process
I am a parent process
I am a parent process
I am a parent process
I am a parent process
I am a parent process
I am a parent process
I am a parent process
I am a parent process
```

8. Write a program that create a child process with the fork () system call. The parent process waits for the child process to finish before printing the contents of the current directory.

Ans:

```
cat: lab3_8: No such file or directory
ubuntu@primary:~$ cat lab3_8.c
#include <stdio.h>
#include <unistd.h>
#include <sys/wait.h>
#include <sys/types.h>
#include <dirent.h>

int main(void) {
  pid_t pid;

  pid = fork();
  if (pid == -1) {
    perror("fork");
    return 1;
  }

  if (pid == 0) {
    printf("Child process: pid=%d\n", getpid());
    // Child process code here
  } else {
    int status;
    printf("Parent process: pid=%d, child pid=%d\n", getpid(), pid);
    waitpid(pid, &status, 0); // Wait for child process to finish
    printf("Child process finished\n");
    // Parent process code here
    DIR *d;
    struct dirent *dir;
    d = opendir(".");
    if (d) {
      while ((dir = readdir(d)) != NULL) {
        printf("%s\n", dir->d_name);
      }
      closedir(d);
    }
  }
  return 0;
}
```

```
ubuntu@primary:~$ ./a.out
Parent process: pid=18351, child pid=18352
Child process: pid=18352
Child process finished
..
```

9. Write a program that create a child process with the fork () system call and print its PID. Following a fork () system call, both parent and child processes print their process type and PID. Additionally, the parent process prints the PID of its child, and the child process prints the PID of its parent.

Ans:

```
ubuntu@primary:~$ cat lab3_9.c
#include <stdio.h>
#include <unistd.h>

int main(void) {
  pid_t pid;

  pid = fork();
  if (pid == -1) {
    perror("fork");
    return 1;
  }

  if (pid == 0) {
    printf("Child process: pid=%d, parent pid=%d\n", getpid(), getppid());
  } else {
    printf("Parent process: pid=%d, child pid=%d\n", getpid(), pid);
  }
  return 0;
}


ubuntu@primary:~$ gcc lab3_9.c
ubuntu@primary:~$ ./a.out
Parent process: pid=18359, child pid=18360
ubuntu@primary:~$ Child process: pid=18360, parent pid=1
```