

## Lab 2

**Name:** Chillara V L N S Pavana Vamsi

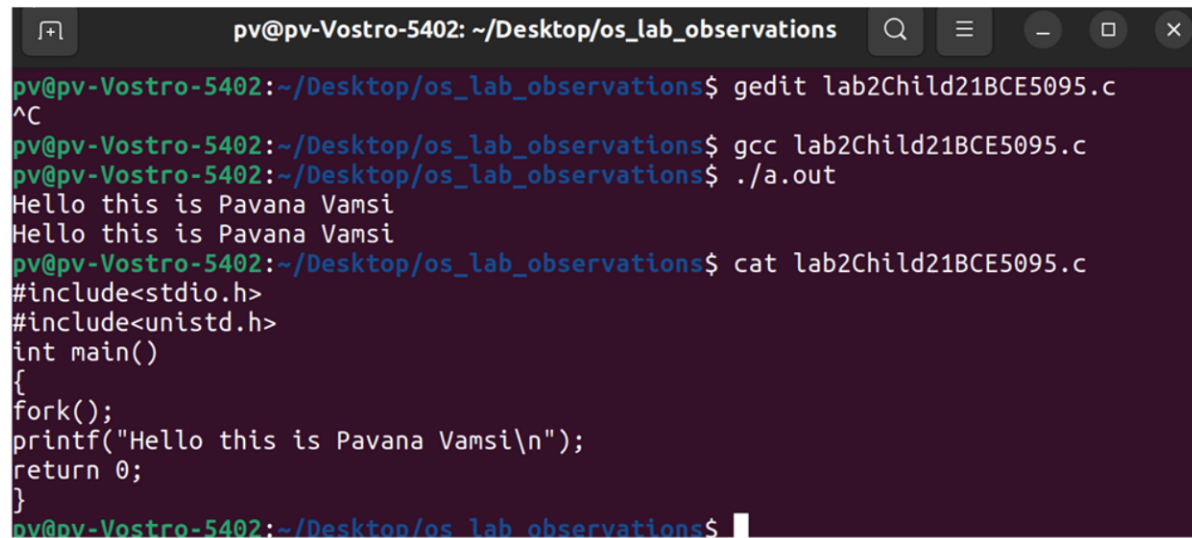
**Reg.no:** 21BCE5095

**Date:** 29/12/2022

**Faculty:** M Sivagami

**L21\_L22\_Lab2 Exercises**

1. Develop a child process to display the hello message, observe the output and give justification of the output.
  - a. When we run the program with one child process, we can see the output as two printf without using any loop. Here the child process created prints the output and other output is print by actual parent that is main ().



```
pv@pv-Vostro-5402: ~/Desktop/os_lab_observations
pv@pv-Vostro-5402:~/Desktop/os_lab_observations$ gedit lab2Child21BCE5095.c
^C
pv@pv-Vostro-5402:~/Desktop/os_lab_observations$ gcc lab2Child21BCE5095.c
pv@pv-Vostro-5402:~/Desktop/os_lab_observations$ ./a.out
Hello this is Pavana Vamsi
Hello this is Pavana Vamsi
pv@pv-Vostro-5402:~/Desktop/os_lab_observations$ cat lab2Child21BCE5095.c
#include<stdio.h>
#include<unistd.h>
int main()
{
fork();
printf("Hello this is Pavana Vamsi\n");
return 0;
}
pv@pv-Vostro-5402:~/Desktop/os_lab_observations$
```

- b. If we use more fork () continusly more child process will be created like if we create n fork at a time then we can observe  $2^n$  hello will be printed in the output screen.



```
pv@pv-Vostro-5402: ~/Desktop/os_lab_observations
pv@pv-Vostro-5402:~/Desktop/os_lab_observations$ gedit lab2Child21BCE5095.c
^C
pv@pv-Vostro-5402:~/Desktop/os_lab_observations$ gcc lab2Child21BCE5095.c
pv@pv-Vostro-5402:~/Desktop/os_lab_observations$ ./a.out
Hello this is Pavana Vamsi
Hello this is Pavana Vamsi
Hello this is Pavana Vamsi
Hello this is Pavana Vamsi
Hello this is Pavana Vamsi
Hello this is Pavana Vamsi
Hello this is Pavana Vamsi
Hello this is Pavana Vamsi
pv@pv-Vostro-5402:~/Desktop/os_lab_observations$
```

## Lab 2

- Develop a child process to display the numbers 500 to 1000 and parent process to display from 1 to 500. Observe the output and give justification for the same.

Here the parent process is done first immediately after parent process done child process started doing its work

```
pv@pv-Vostro-5402: ~/Desktop/os_lab_observations
pv@pv-Vostro-5402:~/Desktop/os_lab_observations$ cat lab2ChildParent21BCE5095.c
#include<stdio.h>
#include<unistd.h>
int main()
{
    int x=fork();
    if (x==0)
    {
        for(int i=500;i<=1000;i++)
            printf("%d ",i);
        printf("\n");
    }
    if(x>0)
    {
        for(int i=1;i<=500;i++)
            printf("%d ",i);
        printf("\n");
    }
    return 0;
}
pv@pv-Vostro-5402:~/Desktop/os_lab_observations$
```

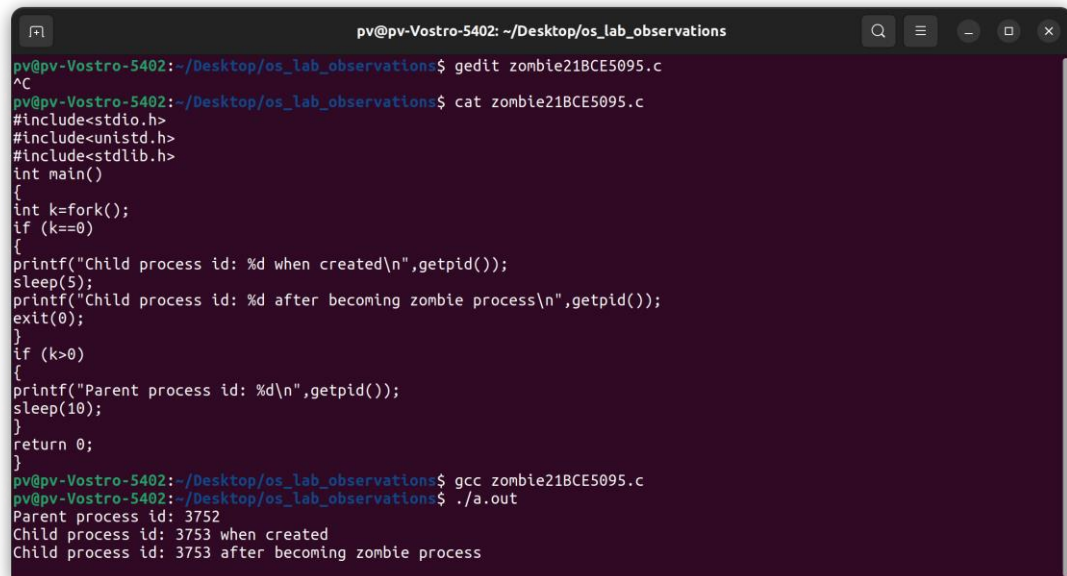
```
pv@pv-Vostro-5402:~/Desktop/os_lab_observations$ ./a.out
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32
33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62
63 64 65 66 67 68 69 70 71 72 73 74 75 76 77 78 79 80 81 82 83 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
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
pv@pv-Vostro-5402:~/Desktop/os_lab_observations$
```

## Lab 2

3. Show the code to create the Zombie and orphan processes with the explanation of how zombie process differs from orphan process.

**a. Zombie process:**

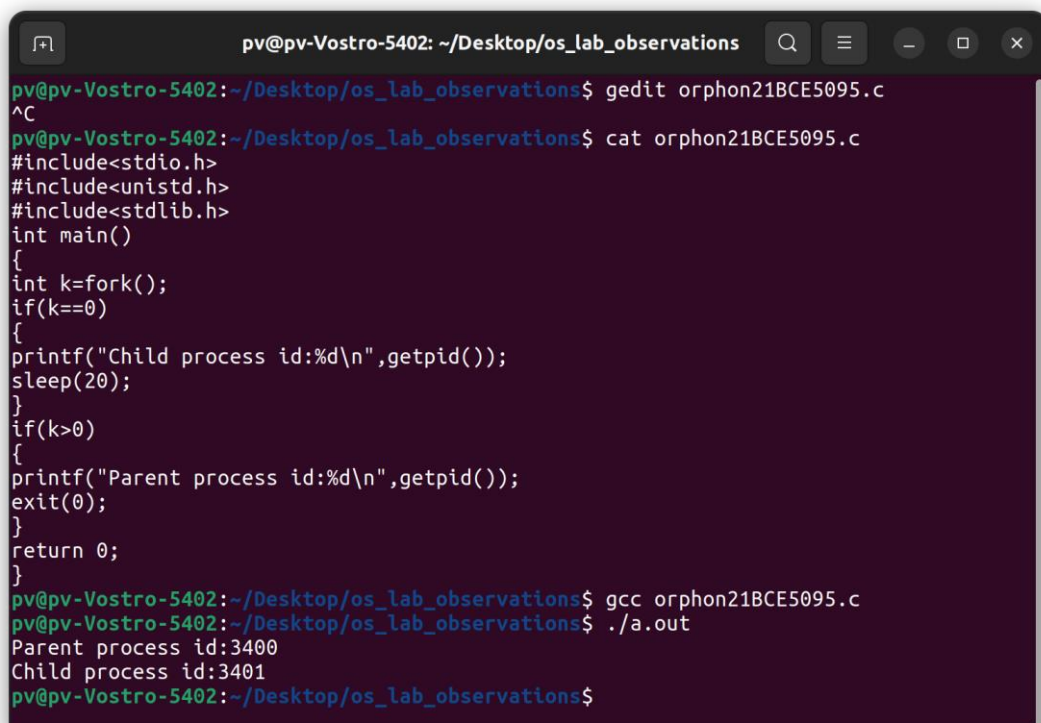
- i. In this child process is done with its work and waiting for the parent, but at that moment parent was unable to kill that child process. So that child process here becomes a zombie process.



```
pv@pv-Vostro-5402: ~/Desktop/os_lab_observations
pv@pv-Vostro-5402:~/Desktop/os_lab_observations$ gedit zombie21BCE5095.c
^C
pv@pv-Vostro-5402:~/Desktop/os_lab_observations$ cat zombie21BCE5095.c
#include<stdio.h>
#include<unistd.h>
#include<stdlib.h>
int main()
{
    int k=fork();
    if (k==0)
    {
        printf("Child process id: %d when created\n",getpid());
        sleep(5);
        printf("Child process id: %d after becoming zombie process\n",getpid());
        exit(0);
    }
    if (k>0)
    {
        printf("Parent process id: %d\n",getpid());
        sleep(10);
    }
    return 0;
}
pv@pv-Vostro-5402:~/Desktop/os_lab_observations$ gcc zombie21BCE5095.c
pv@pv-Vostro-5402:~/Desktop/os_lab_observations$ ./a.out
Parent process id: 3752
Child process id: 3753 when created
Child process id: 3753 after becoming zombie process
```

**b. Orphan Process:**

- i. In this parent process completes its work and exits, but still the child process is doing the work so there will be no one active to kill that child process at the end so it becomes an orphan process.



```
pv@pv-Vostro-5402: ~/Desktop/os_lab_observations
pv@pv-Vostro-5402:~/Desktop/os_lab_observations$ gedit orphon21BCE5095.c
^C
pv@pv-Vostro-5402:~/Desktop/os_lab_observations$ cat orphon21BCE5095.c
#include<stdio.h>
#include<unistd.h>
#include<stdlib.h>
int main()
{
    int k=fork();
    if(k==0)
    {
        printf("Child process id:%d\n",getpid());
        sleep(20);
    }
    if(k>0)
    {
        printf("Parent process id:%d\n",getpid());
        exit(0);
    }
    return 0;
}
pv@pv-Vostro-5402:~/Desktop/os_lab_observations$ gcc orphon21BCE5095.c
pv@pv-Vostro-5402:~/Desktop/os_lab_observations$ ./a.out
Parent process id:3400
Child process id:3401
pv@pv-Vostro-5402:~/Desktop/os_lab_observations$
```



## Lab 2

4. Develop a child process and parent process to count the vowels and consonants accordingly of the given string and display the output.

```
pv@pv-Vostro-5402: ~/Desktop/os_lab_observations
pv@pv-Vostro-5402:~/Desktop/os_lab_observations$ gedit letter21BCE5095.c
^C
pv@pv-Vostro-5402:~/Desktop/os_lab_observations$ gcc letter21BCE5095.c
pv@pv-Vostro-5402:~/Desktop/os_lab_observations$ ./a.out
pavana vamsi
Parent process id:4110
No of consonants found in parent process=3
Child process id:4115
No of vowels found in child process=6
pv@pv-Vostro-5402:~/Desktop/os_lab_observations$
```

```
pv@pv-Vostro-5402: ~/Desktop/os_lab_observations
pv@pv-Vostro-5402:~/Desktop/os_lab_observations$ cat letter21BCE5095.c
#include<stdio.h>
#include<unistd.h>
#include<string.h>
int main()
{
    int k,i,vowelCount=0,consonantCount=0;
    char s[20];
    scanf("%s",s);
    k=fork();
    if(k==0)
    {
        for(i=0;i<strlen(s);i++)
        {
            if(s[i]=='a' || 'e' || 'i' || 'o' || 'u')
                vowelCount++;
        }
        printf("Child process id:%d\nNo of vowels found in child process=%d\n",getpid(),vowelCount);
    }
    if(k>0)
    {
        for(i=0;i<strlen(s);i++)
        {
            if((s[i]>'a' && s[i]<='z') && (s[i]!='a' || 'e' || 'i' || 'o' || 'u'))
                consonantCount++;
        }
        printf("Parent process id:%d\nNo of consonants found in parent process=%d\n",getpid(),consonantCount);
    }
    return 0;
}
pv@pv-Vostro-5402:~/Desktop/os_lab_observations$
```

## Lab 2

- Develop a child process and parent process to perform the matrix addition (by parent process) and matrix subtraction (by child process)

```

pv@pv-Vostro-5402: ~/Desktop/os_lab_observations
pv@pv-Vostro-5402:~/Desktop/os_lab_observations$ gedit matrix21BCE5095.c
^C
pv@pv-Vostro-5402:~/Desktop/os_lab_observations$ gcc matrix21BCE5095.c
pv@pv-Vostro-5402:~/Desktop/os_lab_observations$ ./a.out
Parent process id: 4728
Matrix Addition:
10    10    10
10    10    10
10    10    10
Child process id:4729
Matrix Substraction:
-8    -6    -4
-2     0     2
 4     6     8
pv@pv-Vostro-5402:~/Desktop/os_lab_observations$

```

```

pv@pv-Vostro-5402: ~/Desktop/os_lab_observations
pv@pv-Vostro-5402:~/Desktop/os_lab_observations$ cat matrix21BCE5095.c
#include<stdio.h>
#include<unistd.h>
int main()
{
    int k=fork();
    int a[3][3] = {{1, 2, 3},{4, 5, 6},{7, 8, 9}};
    int b[3][3] = {{9, 8, 7},{6, 5, 4},{3, 2, 1}};
    if(k==0)
    {
        printf("Child process id:%d\n Matrix Substraction:\n",getpid());
        for(int i=0;i<3;i++)
        {
            for(int j=0;j<3;j++)
            {
                printf("%d\t",a[i][j]-b[i][j]);
                printf("\n");
            }
        }
    }
    if(k>0)
    {
        printf("Parent process id: %d\n Matrix Addition:\n",getpid());
        for(int i=0;i<3;i++)
        {
            for(int j=0;j<3;j++)
            {
                printf("%d\t",a[i][j]+b[i][j]);
                printf("\n");
            }
        }
    }
    return 0;
}
pv@pv-Vostro-5402:~/Desktop/os_lab_observations$

```

## Lab 2

6 Develop 3 child processes for doing the below tasks

Child process 1- check the given number is even or odd

Child process 2 - Check whether the given number is prime or not

Child process 3 - Check whether the given number is divisible by 7 or not.

```
pv@pv-Vostro-5402: ~/Desktop/os_lab_observations
pv@pv-Vostro-5402:~/Desktop/os_lab_observations$ cat child3_21BCE5095.c
#include<stdio.h>
#include<unistd.h>
int main()
{
    int n;
    scanf("%d",&n);
    int x=fork();
    if(x==0)
    {
        printf("\n\nChild process 1 id: %d\n",getpid());
        if(n%2==0)
            printf("Given number is even\n");
        else
            printf("Given number is odd\n");
        return 0;
    }
    int y=fork();
    if(y==0)
    {
        int c=0;
        for(int i=2;i<n;i++)
        {
            if(n%i==0)
                c++;
        }
        printf("\n\nChild process 2 id: %d\n",getpid());
        if(c==0)
            printf("Prime Number\n");
        else
            printf("Composite Number\n");
        return 0;
    }
    int z=fork();
    if(z==0)
    {
        printf("\n\nChild process 3 id: %d\n",getpid());
        if(n%7==0)
            printf("Number is divisible by 7\n");
        else
            printf("Number is not divisible by 7\n");
        return 0;
    }
    return 0;
}
pv@pv-Vostro-5402:~/Desktop/os_lab_observations$
```

## Lab 2

```
pv@pv-Vostro-5402: ~/Desktop/os_lab_observations
pv@pv-Vostro-5402:~/Desktop/os_lab_observations$ gedit child3_21BCE5095.c
^C
pv@pv-Vostro-5402:~/Desktop/os_lab_observations$ gcc child3_21BCE5095.c
pv@pv-Vostro-5402:~/Desktop/os_lab_observations$ ./a.out
77

Child process 1 id: 5235
Given number is odd

Child process 2 id: 5236
Composite Number

Child process 3 id: 5237
Number is divisible by 7
pv@pv-Vostro-5402:~/Desktop/os_lab_observations$ ./a.out
43

Child process 1 id: 5240
Given number is odd

Child process 2 id: 5241
Prime Number

Child process 3 id: 5242
Number is not divisible by 7
pv@pv-Vostro-5402:~/Desktop/os_lab_observations$
```