**ALI KASSIM ALI**

**CIT-221-040/2021**

**SYSTEM PROGRAMMING**

**a)**

Dynamic memory allocation enhances memory utilization by allocating memory during runtime based on the program's needs. This avoids wastage caused by preallocating memory that may not be fully used. For example:

* Efficient Utilization: Memory is allocated only when needed.
* Flexibility: Applications can request or release memory as required, reducing fragmentation.
* Optimal Resource Sharing: Unused memory is returned to the system, making it available for other processes.

**b)**

char vech[] = "BUS"; // Defining vech as "BUS"

int n = 10;

**c)**

1. **Static Allocation**

Memory is allocated at compile-time and remains fixed throughout the program's execution. This allocation is typically used for global and static variables.

**ii. Stack Allocation**

Memory is allocated for local variables and function calls in a stack-like manner, using a Last-In-First-Out (LIFO) structure. Memory is automatically released when a function call ends.

**iii. Heap Allocation**

Memory is allocated dynamically at runtime from the heap, a large pool of memory reserved for long-term or flexible use.

**d)**

chmod 751 filename

7: Full access for the user.

5: Read and execute permission for the group.

1: Execute permission for others.

**e) Virtual Memory**

Virtual memory extends physical memory by using disk space to simulate additional RAM.

* **Enhancements**:
  1. Allows running larger programs than the available physical memory.
  2. Supports multitasking by isolating memory spaces for different processes.
  3. Provides protection and security through memory isolation.

**f) Paging in Modern Operating Systems**

Paging divides memory into fixed-size blocks (pages).

* **How it Works**:
  1. Logical memory is divided into pages, and physical memory is divided into frames.
  2. Pages are mapped to frames using a page table.
  3. Reduces fragmentation by utilizing all memory blocks.
* **Advantages**:
  1. Efficient memory management.
  2. Facilitates virtual memory implementation.

**g)**

chmod 751 filename

7: Full access for the user.

5: Read and execute permission for the group.

1: Execute permission for others.

**h)**

**i)**

#include <stdio.h>

int main() {

int n, i;

unsigned long long factorial = 1;

printf("Enter a number: ");

scanf("%d", &n);

if (n < 0)

printf("Factorial of negative number doesn't exist.");

else {

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

factorial \*= i;

}

printf("Factorial of %d = %llu\n", n, factorial);

}

return 0;

}

ii)

compile the program - gcc factorial.c -o factorial

run the program - ./factorial

i)

shapes=("square")

echo "Shape: ${shapes[0]}"

**j)**

i) echo $((6 + 3))

ii) echo $((20 % 3))

iii) echo $((10 \* 3))

**k)** ls > directory\_list.txt