

PROCESSI

WIN-API	BOOL CreateProcess()
DESCRIZIONE	Crea un nuovo processo (child).
	<ul style="list-style-type: none"> • LPCTSTR lpApplicationName → puntatore a una stringa di caratteri (LPCTSTR = long pointer to character string) che indica il nome del programma che deve essere lanciato all'interno del nuovo processo. • LPTSTR lpCommandLine → puntatore a una stringa che identifica l'insieme degli argomenti del main dell'applicazione che viene a essere lanciata nell'ambito del nuovo processo. • LPSECURITY_ATTRIBUTES lpProcessAttributes → puntatore alla struttura SECURITY_ATTRIBUTES che specifica la gestione della sicurezza del contenitore di memoria dell'applicazione che viene a essere lanciata. • LPSECURITY_ATTRIBUTES lpThreadAttributes → puntatore alla struttura SECURITY_ATTRIBUTES che specifica la gestione della sicurezza del thread (percorso di esecuzione) che viene attivato.
PARAMETRI	<ul style="list-style-type: none"> • BOOL bInheritHandles → booleano che indica se è possibile far ereditare al nuovo processo gli handle del parent. • DWORD dwCreationFlags → valore che specifica alcune informazioni su ciò che il Kernel dovrà fare quando gestirà il nuovo processo (e.g. priorità dell'applicazione). • LPVOID lpEnvironment → puntatore generico a una locazione di memoria dove si trovano le informazioni ambientali della nuova applicazione. • LPCTSTR lpCurrentDirectory → puntatore che indica la PWD del nuovo processo. • LPSTARTUPINFO lpStartupInfo → puntatore alla struttura STARTUPINFO. • LPPROCESS_INFORMATION lpProcessInformation → puntatore alla struttura PROCESS_INFORMATION.
OUTPUT	Un valore diverso da zero in caso di successo, 0 altrimenti.

WIN-API	DWORD WINAPI GetCurrentProcessId()
DESCRIZIONE	Richiude al Kernel il codice numerico del processo corrente.
PARAMETRI	/
OUTPUT	Codice numerico del processo corrente

WIN-API	DWORD WINAPI GetProcessId()
DESCRIZIONE	Richiude al Kernel il codice numerico di un altro processo. • <u>_In_</u> HANDLE Process → maniglia verso l'oggetto di cui si vuole acquisire l'id.
PARAMETRI	"_In_" indica che il parametro è da dare in input al Kernel il quale non può generare side-effect in memoria, bensì può solo leggere (al contrario di " <u>_Out_</u> ").
OUTPUT	Codice numerico del processo di cui è stato specificato l'handle

WIN-API	VOID ExitProcess()
DESCRIZIONE	Richiude al Kernel la terminazione del processo chiamante.
PARAMETRI	• UINT uExitCode → codice numerico con cui si vuole terminare
OUTPUT	/

WIN-API	DWORD WaitForSingleObject()
DESCRIZIONE	Richiede la permanenza nello stato di blocco in attesa che un determinato oggetto sia disponibile per il chiamante (se l'oggetto è un processo, ciò avviene quando esso termina).
PARAMETRI	• HANDLE hHandle → maniglia verso l'oggetto che si sta attendendo • DWORD dwMilliseconds → timeout dopo cui il chiamante riprenderà il controllo a prescindere dall'esito dell'attesa.
OUTPUT	WAIT_FAILED in caso di fallimento (ovvero nel caso in cui l'oggetto non si sia reso disponibile entro il timeout)

WIN-API	<code>int GetExitCodeProcess()</code>
DESCRIZIONE	Richiede il codice di terminazione di uno specifico processo.
PARAMETRI	<ul style="list-style-type: none"> - HANDLE hProcess → maniglia verso il processo di cui si vuole acquisire il codice di terminazione - LPDWORD lpExitCode → puntatore a una cella di memoria in cui il Kernel restituirà il codice di terminazione
OUTPUT	<code>STILL_ACTIVE</code> se il processo target è ancora attivo

WIN-API	<code>BOOL WINAPI TerminateProcess()</code>
DESCRIZIONE	Chiede di far terminare uno specifico processo di cui si possiede l'handle.
PARAMETRI	<ul style="list-style-type: none"> - In - HANDLE hProcess → maniglia verso il processo di cui si richiede la terminazione - In - UNIT uExitCode → codice di terminazione del processo che viene interrotto
OUTPUT	Un valore diverso da zero in caso di successo, 0 altrimenti

WIN-API	<code>LPTCH WINAPI GetEnvironmentStrings()</code>
DESCRIZIONE	Richiede il valore di tutte le variabili d'ambiente.
PARAMETRI	/
OUTPUT	Puntatore alla prima delle stringhe che identificano le variabili d'ambiente; tali stringhe sono organizzate in una sequenza di token che termina con la stringa nulla (ovvero con un doppio '\0').

WIN-API	<code>DWORD WINAPI GetEnvironmentVariable()</code>
DESCRIZIONE	Richiude il valore di una specifica variabile d'ambiente. • <u>In_opt_</u> <code>LPCTSTR lpName</code> → nome della variabile d'ambiente. • <u>-opt-</u> indica che il parametro è opzionale.
PARAMETRI	• <u>Out_opt_</u> <code>LPTSTR lpBuffer</code> → puntatore al buffer in cui il valore richiesto verrà scritto. • <u>In_</u> <code>DWORD nSize</code> → spazio massimo che può essere occupato nel buffer di cui è stato passato il puntatore.
OUTPUT	Valore della variabile d'ambiente specificata

WIN-API	<code>BOOL WINAPI SetEnvironmentVariable()</code>
DESCRIZIONE	Setta il valore di una specifica variabile d'ambiente.
PARAMETRI	• <u>In_</u> <code>LPCTSTR lpName</code> → nome della variabile d'ambiente • <u>In_opt_</u> <code>LPTSTR lpValue</code> → valore della variabile d'ambiente
OUTPUT	Un valore diverso da zero in caso di successo, 0 altrimenti

WIN-API	<code>BOOL WINAPI FreeEnvironmentStrings()</code>
DESCRIZIONE	Elimina tutte le variabili d'ambiente.
PARAMETRI	• <u>In_</u> <code>LPTCH lpszEnvironmentBlock</code> → puntatore alla prima delle stringhe che identificano le variabili d'ambiente
OUTPUT	Un valore diverso da zero in caso di successo, 0 altrimenti

WIN-API	HANDLE CreateThread()
DESCRIZIONE	<p>Crea un nuovo thread all'interno della stessa applicazione del chiamante, in modo tale che le due tracce condivideranno il medesimo address space.</p> <ul style="list-style-type: none"> • LPSECURITY_ATTRIBUTES lpThreadAttributes → puntatore al = la struttura SECURITY_ATTRIBUTES • SIZE_T dwStackSize → dimensione dello stack per il nuovo thread • LPTHREAD_START_ROUTINE lpStartAddress → puntatore al = l'indirizzo di memoria della sezione testo da cui il nuovo thread deve iniziare la sua esecuzione. • LPVOID lpParameter → puntatore generico da passare come parametro di input al nuovo thread tramite un registro di CPU. Esso può puntare a un qualunque indirizzo di memoria per leggere qualunque informazione. • DWORD dwCreationFlags → intero che dà informazioni di base sullo scheduling (e.g. quanto è prioritario il nuovo thread rispetto agli altri). • LPDWORD lpThreadId → puntatore al codice numerico del nuovo thread
PARAMETRI	
OUTPUT	Un handle al nuovo thread in caso di successo, INVALID_HANDLE_VALUE (o NULL) altrimenti.

WIN-API	VOID ExitThread()
DESCRIZIONE	Richiede al Kernel la terminazione del thread chiamante.
PARAMETRI	<ul style="list-style-type: none"> • DWORD dwExitCode → codice d'uscita del thread che viene interrotto
OUTPUT	/

WIN-API	<code>int GetExitCodeThread()</code>
DESCRIZIONE	Richiede il codice di terminazione di uno specifico thread. • HANDLE hThread → maniglia verso il thread di cui si vuole acquisire il codice di terminazione.
PARAMETRI	• LPDWORD lpExitCode → puntatore all'area di memoria dove verrà scritto il codice di uscita richiesto.
OUTPUT	0 in caso di fallimento

CPU-SCHEDULING

WIN-API	BOOL WINAPI SetPriorityClass()
DESCRIZIONE	Imposta la classe di priorità di un processo.
PARAMETRI	<ul style="list-style-type: none">• In- HANDLE hProcess• In- DWORD dwPriorityClass → classe di priorità da assegnare al processo di cui viene passata la maniglia.
OUTPUT	

WIN-API	BOOL WINAPI SetThreadPriority()
DESCRIZIONE	Imposta la priorità di riferimento (quella minima) di un thread.
PARAMETRI	<ul style="list-style-type: none">• In- HANDLE hThread• In- int nPriority → priorità di riferimento da assegnare al thread di cui viene passata la maniglia. <p>NB: Diziente anche della classe di priorità del processo in cui questa traccia vive.</p>
OUTPUT	

WIN-API	BOOL WINAPI SetProcessAffinityMask()
DESCRIZIONE	Imposta l'affinità di uno specifico processo nella regola di scheduling; le affinità dei thread che vivono in quel processo dovranno necessariamente essere dei sottinsiemi.
PARAMETRI	<ul style="list-style-type: none">• In- HANDLE hProcess• In- DWORD_PTR dwProcessAffinityMask → puntatore alla maschera di bit che indica le CPU permesse per il processo di cui viene passata la maniglia. <p>NB: Questo pointer punta a una DWORD, che rappresenta solo valori a 64 bit; se si hanno più di 64 processori, ogni bit è relativo a un intero gruppo di 64 CPU.</p>
OUTPUT	

WIN-API	DWORD_PTR WINAPI SetThreadAffinityMask()
DESCRIZIONE	Imposta l'affinità di uno specifico thread nella regola di scheduling.
PARAMETRI	<ul style="list-style-type: none">• In - HANDLE hThread• In - DWORD_PTR dwThreadAffinityMask
OUTPUT	

VIRTUAL FILE SYSTEM

WIN-API	HANDLE CreateFile()
DESCRIZIONE	Crea un nuovo file.
	<ul style="list-style-type: none">• LPCTSTR lpFileName → puntatore alla stringa che definisce il nome del file da creare (non è specificata la codifica ASCII/UNICODE).• DWORD dwDesiredAccess → specifica i permessi di accesso relativi solo alla sessione di I/O che sta per essere istanziata (e.g. GENERIC_READ, GENERIC_WRITE).• DWORD dwShareMode → specifica se più sessioni diverse possono condividere il file (e.g. FILE_SHARE_READ, FILE_SHARE_WRITE, FILE_SHARE_DELETE).• LPSECURITY_ATTRIBUTES lpSecurityAttributes → puntatore alla struttura SECURITY_ATTRIBUTES che specifica la gestione della sicurezza del file. Il secondo parametro di questa struttura (LPVOID lpSecurityDescriptor) è un puntatore a una ACL.• DWORD dwCreationDisposition → specifica l'azione da fare se il file esisteva già e quella da fare se non esisteva (e.g. CREATE_NEW, CREATE_ALWAYS, OPEN_EXISTING, TRUNCATE_EXISTING).• DWORD dwFlagsAndAttributes → specifica caratteristiche varie del file (e.g. FILE_ATTRIBUTES_NORMAL, ...HIDDEN, ...DELETE_ON_CLOSE).• HANDLE hTemplateFile → handle a un file di template, il cui contenuto verrà riassegnato nel file da creare.
PARAMETRI	
OUTPUT	Handle al nuovo file in caso di successo

WIN-API	DWORD WINAPI GetSecurityInfo()
	Richiude la ACL di uno specifico oggetto di cui si conosce la maniglia.
	Win-API correlate:
	→ GetNamedSecurityInfo(), che richiude la ACL di uno specifico oggetto di cui si conosce il nome.
DESCRIZIONE	→ SetSecurityInfo(), che modifica la ACL di uno specifico oggetto di cui si conosce la maniglia.
	→ SetNamedSecurityInfo(), che modifica la ACL di uno specifico oggetto di cui si conosce il nome.
	<ul style="list-style-type: none"> • In - HANDLE handle • In - SE_OBJECT_TYPE ObjectType • In - SECURITY_INFORMATION SecurityInfo → specifica quale informazione deve essere restituita. • Out_opt - PSID *ppsidOwner → puntatore a puntatore al l'area di memoria in cui dovrà essere registrato il SID del proprietario dell'oggetto di cui viene richiesta la ACL. • Out_opt - PSID *ppsidGroup • Out_opt - PACL *ppDacl • Out_opt - PACL *ppSacl • Out_opt - PSECURITY_DESCRIPTOR *ppSecurityDescriptor → puntatore all'area di memoria in cui dovranno essere restituite la ACL ed eventualmente i SID, la Dacl e la Sacl dell'oggetto di cui viene passato l'handle come parametro.
PARAMETRI	
OUTPUT	

WIN-API	BOOL WINAPI LookupAccountSid()
DESCRIZIONE	Richiude informazioni riguardanti l'utenza associata a un identificato SID (e.g. il nome).
PARAMETRI	<ul style="list-style-type: none"> - In-opt- LPCTSTR lpSystemName - In- PSID lpSid → puntatore all'area di memoria contenente il SID - Out-opt- LPTSTR lpName → puntatore all'area di memoria in cui dovrà essere registrato il nome dell'utente associato al SID. - Imout- LPDWORD cchName → dimensione dell'area di memoria, in cui dovrà essere registrato il nome dell'utente associato al SID. - Out-opt- LPTSTR lpReferenceDomainName - Imout- LPDWORD cchReferenceDomainName - Out- PSID_NAME_USE flags
OUTPUT	

WIN-API	BOOL CloseHandle()
DESCRIZIONE	Chiude un canale di I/O verso uno specifico file.
PARAMETRI	<ul style="list-style-type: none"> - HANDLE hObject
OUTPUT	0 in caso di fallimento

WIN-API	BOOL ReadFile()
DESCRIZIONE	Legge un file.
	<ul style="list-style-type: none"> • HANDLE hFile • LPVOID lpBuffer → puntatore al buffer in cui si vuole acquisire i dati letti; • DWORD nNumberOfBytesToRead → numero massimo di byte da leggere
PARAMETRI	<ul style="list-style-type: none"> • LPDWORD lpNumberOfBytesRead → puntatore all'area di memoria in cui dovrà essere registrato il numero di byte effettivamente letti. • LPOVERLAPPED lpOverlapped → puntatore a una struttura dati che indica se la restituzione dei byte letti sul buffer deve essere sincrona con l'operazione di lettura o meno.
OUTPUT	0 in caso di fallimento

WIN-API	BOOL WriteFile()
DESCRIZIONE	Scrive su un file.
	<ul style="list-style-type: none"> • HANDLE hFile • LPCVOID lpBuffer
PARAMETRI	<ul style="list-style-type: none"> • DWORD nNumberOfBytesToWrite • LPDWORD lpNumberOfBytesWritten • LPOVERLAPPED lpOverlapped
OUTPUT	0 in caso di fallimento

WIN-API	BOOL DeleteFile()
DESCRIZIONE	Rimuove un hard link di uno specifico file.
PARAMETRI	• LPCTSTR lpFileName
OUTPUT	Un valore diverso da zero in caso di successo, 0 altrimenti.

WIN-API	<code>DWORD SetFilePointer()</code>
DESCRIZIONE	Riposiziona il file pointer.
PARAMETRI	<ul style="list-style-type: none"> • <code>HANDLE hFile</code> • <code>LONG lDistanceToMove</code> → i 32 bit meno significativi di un intero (positivo o negativo) che indica il numero di caratteri di cui viene spostato il file pointer (in avanti o all'interno a seconda del segno) a partire dal punto indicato dal parametro <code>dwMoveMethod</code>. • <code>PULONG lpDistanceToMoveHigh</code> → puntatore a un long contenente i 32 bit più significativi del valore in <code>lDistanceToMove</code>. • <code>DWORD dwMoveMethod</code> → valore che indica l'opzione di spostamento: dall'inizio (<code>FILE_BEGIN</code>), dalla posizione corrente (<code>FILE_CURRENT</code>) o dalla fine (<code>FILE_END</code>).
OUTPUT	I 32 bit meno significativi del nuovo valore del file pointer (valutato dall'inizio del file) in caso di successo, <code>INVALID_SET_FILE_POINTER</code> altrimenti.

WIN-API	<code>HANDLE WINAPI GetStdHandle()</code>
DESCRIZIONE	Richiede l'handle verso l'oggetto associato a uno specifico canale standard.
PARAMETRI	<ul style="list-style-type: none"> • In: <code>DWORD nStdHandle</code> → canale standard (<code>STD_IN_HANDLE</code>, <code>STD_OUTPUT_HANDLE</code> oppure <code>STD_ERROR_HANDLE</code>) di cui si vuole conoscere l'handle associato.
OUTPUT	Handle verso l'oggetto associato al canale standard specificato.

WIN-API	<code>BOOL WINAPI SetStdHandle()</code>
DESCRIZIONE	Assegna un nuovo handle a uno specifico canale standard.
PARAMETRI	<ul style="list-style-type: none"> • In <code>DWORD mStdHandle</code> • In <code>HANDLE hHandle</code>
OUTPUT	
WIN-API	<code>BOOL WINAPI CreateDirectory()</code>
DESCRIZIONE	Crea una directory.
PARAMETRI	<ul style="list-style-type: none"> • In <code>LPCTSTR lpPathName</code> → nome della nuova directory • In opt <code>LPSECURITY_ATTRIBUTES lpSecurityAttributes</code>
OUTPUT	
WIN-API	<code>BOOL WINAPI RemoveDirectory()</code>
DESCRIZIONE	Rimuove una directory.
PARAMETRI	<ul style="list-style-type: none"> • In <code>LPCTSTR lpPathName</code>
OUTPUT	
WIN-API	<code>BOOL WINAPI CreateHardLink()</code>
DESCRIZIONE	Crea un nuovo hard link per uno specifico file, instantaneamente una nuova entry nella directory corrente e un nuovo elemento della MFT associato a questo file.
PARAMETRI	<ul style="list-style-type: none"> • <code>LPCTSTR lpFileName</code> → nuovo nome per il file • <code>LPCTSTR lpExistingFileName</code> → nome già esistente del file • <code>LPSECURITY_ATTRIBUTES lpSecurityAttributes</code>
OUTPUT	

WIN-API	BOOLEAN WINAPI CreateSymbolicLink()
DESCRIZIONE	Creare un nuovo soft link per uno specifico file.
PARAMETRI	<ul style="list-style-type: none"> • -- in LPTSTR lpSymlinkFileName → nuovo nome per il file • -- in LPTSTR lpTargetFileName → nome già esistente del file • -- in DWORD dwFlags → indica se lpTargetFileName è una directory.
OUTPUT	

WIN-API	BOOL FlushFileBuffers()
DESCRIZIONE	Richiede che qualunque dato aggiornato di uno specifico file venga riportato sul dispositivo di memoria di massa e porta il thread chiamante nello stato di blocco finché questa operazione non verrà completata.
PARAMETRI	- HANDLE hFile
OUTPUT	

COMUNICAZIONE TRA THREAD / PROCESSI

WIN-API	BOOL CreatePipe()
DESCRIZIONE	Crea una PIPE.
PARAMETRI	<ul style="list-style-type: none">• PHANDLE hReadPipe → puntatore a una variabile in cui verrà scritto l'handle verso l'estremo di lettura della PIPE• PHANDLE hWritePipe → puntatore a una variabile in cui verrà scritto l'handle verso l'estremo di lettura della PIPE• LPSECURITY_ATTRIBUTES lpPipeAttributes• DWORD nSize → dimensione del buffer della nuova PIPE
OUTPUT	0 in caso di fallimento

WIN-API	HANDLE CreateNamedPipe()
DESCRIZIONE	Crea una named PIPE.
	<ul style="list-style-type: none"> • LPCTSTR lpName → percorso nome della named PIPE che deve avere le seguenti caratteristiche: <ul style="list-style-type: none"> → Deve iniziare col preambolo " \\.\pipe " → <u>\\.\pipe</u> → Il nome effettivo che segue immediatamente il preambolo deve avere una lunghezza inferiore ai 256 caratteri e non deve includere i simboli "\ " ":" . • DWORD dwOpenMode → può assumere uno dei seguenti tre valori: <ul style="list-style-type: none"> → PIPE_ACCESS_INBOUND : il thread creatore può utilizzare la named PIPE solo per leggere dati dal buffer. → PIPE_ACCESS_OUTBOUND : il thread creatore può utilizzare la named PIPE solo per scrivere dati sul buffer. → PIPE_ACCESS_DUPLEX : la named PIPE creata è bidirezionale, ha cioè due buffer distinti: il thread creatore può utilizzare un buffer solo per le operazioni di scrittura e l'altro solo per le operazioni di lettura.
PARAMETRI	<ul style="list-style-type: none"> • DWORD dwPipeMode → modalità con cui si vuole lavorare sulla named PIPE. Alcuni esempi sono: <ul style="list-style-type: none"> → PIPE_TYPE_BYTE : la comunicazione avviene secondo un modello stream. → PIPE_TYPE_MESSAGE : la comunicazione avviene a blocchi. • DWORD nMaxInstances → numero massimo di istanze (named PIPE con lo stesso nome) che possono essere create. • DWORD nOutBufferSize → dimensione del buffer di output • DWORD nInBufferSize → dimensione del buffer di input • DWORD nDefaultTimeOut → timeout per le operazioni bloccanti • LPSECURITY_ATTRIBUTES lpSecurityAttributes
OUTPUT	Handle verso la nuova named PIPE in caso di successo, INVALID_HANDLE_VALUE altrimenti

WIN-API	HANDLE CreateMailslot()
DESCRIZIONE	Crea un mailslot.
PARAMETRI	<ul style="list-style-type: none"> • LPCTSTR lpName → nome del mailslot che deve iniziare col preambolo "\.\mailslot\". • DWORD nMaxMessageSize → taglia massima dei messaggi che potranno essere depositati. • DWORD lReadTimeout → timeout per i servizi bloccanti. • LPSECURITY_ATTRIBUTES lpSecurityAttributes
OUTPUT	Handle al nuovo mailslot in caso di successo, INVALID_HANDLE_VALUE altrimenti.

GESTIONE DELLA MEMORIA

WIN-API	<code>LPVOID VirtualAlloc()</code>
DESCRIZIONE	Mappa delle pagine contigue all'interno dell'address space del processo chiamante. <small>(non visibili)</small>
	<ul style="list-style-type: none"> • <code>LPVOID lpAddress</code> → punto del contenitore di memoria in cui inizia la regione da mappare. • <code>SIZE_T dwSize</code> → quantità di memoria da mappare. • <code>DWORD flAllocationType</code> → opzioni sulla mappatura da effettuare. Alcuni esempi sono: <ul style="list-style-type: none"> → <code>MEM_COMMIT</code>: le pagine saranno utilizzabili. → <code>MEM_RESERVE</code>: le pagine non saranno utilizzabili. → <code>MEM_RESET</code>: le pagine, che erano già in memoria RAM, non sono più di interesse e vengono dematerializzate. → <code>MEM_RESET_UNDO</code>: viene annullata l'operazione di <code>RESET</code>. Se possibile, le pagine che prima erano state dematerializzate, vengono riportate in RAM all'interno degli stessi frame. • <code>DWORD flProtect</code> → permessi di accesso per le pagine da mappare
PARAMETRI	
OUTPUT	

WIN-API	<code>BOOL VirtualFree()</code>
DESCRIZIONE	De-mappa o dematerializza delle pagine contigue all'interno dell'address space del processo chiamante.
	<ul style="list-style-type: none"> • <code>LPVOID lpAddress</code> • <code>SIZE_T dwSize</code> • <code>DWORD dwFreeType</code> → flag che può assumere uno dei seguenti due valori: <ul style="list-style-type: none"> → <code>MEM_RELEASE</code>: le pagine verranno de-mappate. → <code>MEM_DECOMMIT</code>: le pagine verranno solo dematerializzate, ma non saranno più reutilizzabili.
PARAMETRI	
OUTPUT	

WIN-API	BOOL VirtualProtect()
DESCRIZIONE	Modifica i permessi di accesso di alcune pagine.
PARAMETRI	<ul style="list-style-type: none"> • LPVOID lpAddress • DWORD dwSize • DWORD flNewProtect → nuove regole di protezione • PDWORD lpflOldProtect → puntatore all'area di memoria contenente le vecchie regole di protezione
OUTPUT	

WIN-API	HANDLE CreateFileMapping()
DESCRIZIONE	Invoca un file mapping in memoria (che può essere condivisa).
PARAMETRI	<ul style="list-style-type: none"> • HANDLE hFile → handle del file il cui contenuto deve essere mappato all'interno della memoria condivisa • LPSECURITY_ATTRIBUTES lpAttributes • DWORD flProtect → permessi di accesso al file mapping • DWORD dwMaximumSizeHigh → i 32 bit più significativi della dimensione massima del file mapping • DWORD dwMaximumSizeLow → i 32 bit meno significativi della dimensione massima del file mapping • LPCTSTR lpName → nome del file mapping
OUTPUT	Handle al file mapping

WIN-API	HANDLE OpenFileMapping()
DESCRIZIONE	apre un file mapping esistente.
PARAMETRI	<ul style="list-style-type: none"> • DWORD dwDesiredAccess → modalità d'accesso • BOOL bInheritHandle → booleano che indica se l'handle restituito deve essere ereditato o meno da processi figli. • LPCTSTR lpName
OUTPUT	Handle al file mapping

WIN-API	<code>LPVOID MapViewOfFile()</code>
DESCRIZIONE	Aggiorna un file mapping a determinate pagine logiche del processo chiamante.
PARAMETRI	<ul style="list-style-type: none"> • <code>HANDLE hFileMappingObject</code> → maniglia del file mapping • <code>DWORD dwDesiredAccess</code> • <code>DWORD dwFileOffsetHigh</code> → i 32 bit più significativi dell'offset del file mapping da cui deve iniziare il collegamento. • <code>DWORD dwFileOffsetLow</code> → i 32 bit meno significativi dell'offset del file mapping da cui deve iniziare il collegamento. • <code>SIZE_T dwNumberOfBytesToMap</code> → numero di byte del file mapping che devono essere collegati.
OUTPUT	Puntatore all'area di memoria contenente il file mapping in caso di successo, -1 altrimenti.

WIN-API	<code>BOOL UnmapViewOfFile()</code>
DESCRIZIONE	Sgancia un file mapping da determinate pagine logiche del processo chiamante.
PARAMETRI	<ul style="list-style-type: none"> • <code>LPVOID lpBaseAddress</code> → puntatore all'area di memoria contenente il file mapping
OUTPUT	<code>FALSE</code> in caso di fallimento

SINCRONIZZAZIONE

WIN-API	UCHAR WINAPI InterlockedBitTestAndSet()
DESCRIZIONE	Consulta un particolare bit: se è uguale a 0 allora lo aggior= na a 1, altrimenti non effettua altre operazioni. Mentre questa funzione lavora, viene attivato un meccanismo che blocca qualunque thread voglia accedere a questo stesso bit. Win-API correlate: → BitTestAndSet(), che funziona come InterlockedBitTest- AndSet() ma senza il meccanismo di protezione che bloc- ca eventuali altri thread.
PARAMETRI	- In - LONG volatile *Base → puntatore a una locazione di me- moria • - In - LONG Bit → bit su cui effettuare il TEST AND SET
OUTPUT	

WIN-API	HANDLE CreateMutex()
DESCRIZIONE	Crea un mutex.
PARAMETRI	<ul style="list-style-type: none">• LPSECURITY_ATTRIBUTES lpMutexAttributes• BOOL bInitialOwner → booleano che indica se inizialmente il mutex deve essere disponibile o meno.• LPCTSTR lpName → nome del mutex
OUTPUT	Handle al nuovo mutex in caso di successo, NULL altrimenti.

WIN-API	HANDLE OpenMutex()
DESCRIZIONE	Apri un mutex.
PARAMETRI	<ul style="list-style-type: none"> • DWORD dwDesiredAccess → tipo di accesso che si vuole effettuare • BOOL bInheritHandle → booleano che indica se la maniglia è ereditabile o meno • LPCTSTR lpName
OUTPUT	Handle al mutex in caso di successo, NULL altrimenti

WIN-API	DWORD WaitForSingleObject()
DESCRIZIONE	Accede a un mutex/semaforo dopo aver eventualmente atteso la sua disponibilità.
PARAMETRI	<ul style="list-style-type: none"> • HANDLE hHandle • DWORD dwMilliseconds → timeout dopo il quale il chiamante esce a priori dallo stato di wait.
OUTPUT	WAIT_FAILED in caso di fallimento

WIN-API	BOOL ReleaseMutex()
DESCRIZIONE	Rilascia un mutex.
PARAMETRI	<ul style="list-style-type: none"> • HANDLE hMutex
OUTPUT	0 in caso di fallimento

WIN-API	HANDLE CreateSemaphore()
DESCRIZIONE	Crea un semaforo.
PARAMETRI	<ul style="list-style-type: none"> • LPSECURITY_ATTRIBUTES lpSemaphoreAttributes • LONG lInitialCount → numero di token che devono essere inizialmente disponibili • LONG lMaximumCount → numero massimo di token che il nuovo semaforo può contenere • LPCTSTR lpName
OUTPUT	Handle al nuovo semaforo in caso di successo, NULL altrimenti.

WIN-API	HANDLE OpenSemaphore()
DESCRIZIONE	Apri un semaforo.
PARAMETRI	<ul style="list-style-type: none"> • DWORD dwDesiredAccess • BOOL bInheritHandle • LPCTSTR lpName
OUTPUT	Handle al semaforo in caso di successo, NULL altrimenti.

WIN-API	BOOL ReleaseSemaphore()
DESCRIZIONE	Rilascia alcuni token a un semaforo.
PARAMETRI	<ul style="list-style-type: none"> • HANDLE hSemaphore • LONG lReleaseCount → numero di token da rilasciare. • LPVOID lpPreviousCount → puntatore all'area di memoria in cui verrà inserito il vecchio valore del semaforo.
OUTPUT	

WIN-API	DWORD WINAPI WaitForMultipleObjects()
DESCRIZIONE	Richiede la permanenza nello stato di blocco in attesa della disponibilità di più oggetti.
PARAMETRI	<ul style="list-style-type: none"> • In - DWORD nCount → numero di oggetti per cui viene invocata l'attesa • In - const HANDLE *lpHandles → array di maniglie agli oggetti per cui viene invocata l'attesa • In - BOOL bWaitAll → booleano che indica se il chiamante deve attendere la disponibilità di tutti gli oggetti di cui viene passata la maniglia oppure solo del primo che si sbloccherà. • In - DWORD dwMilliseconds
OUTPUT	

EVENTI

WIN-API	BOOL WINAPI SetConsoleCtrlHandler()
DESCRIZIONE	Manipla la pila degli handler.
PARAMETRI	<ul style="list-style-type: none">- In - opt - PHANDLER_ROUTINE HandlerRoutine → puntatore a uno specifico gestore- In - Bool Add → booleano che, se posto a TRUE, aggiunge il gestore alla pila, altrimenti lo rimuove.
OUTPUT	0 in caso di fallimento

WIN-API	BOOL WINAPI GenerateConsoleCtrlEvent()
DESCRIZIONE	Invia un segnale a un altro processo o gruppo di processi.
PARAMETRI	<ul style="list-style-type: none">- In - DWORD dwCtrlEvent → codice numerico del segnale da inviare- In - DWORD dwProcessGroupId → identificatore del processo (o gruppo di processi) che deve ricevere il segnale
OUTPUT	

WIN-API	ATOM RegisterClass()
DESCRIZIONE	Crea una classe di finestre.
PARAMETRI	<ul style="list-style-type: none">- const WNDCLASS *lpWndClass → puntatore a una struttura rea contenente i metadati di base relativi alla nuova classe di finestre.
OUTPUT	0 in caso di fallimento

WIN-API	<code>HWND CreateWindow()</code>
DESCRIZIONE	Crea una finestra ma non la visualizza sullo schermo.
PARAMETRI	<ul style="list-style-type: none"> • <code>LPCTSTR lpClassName</code> → nome della classe a cui deve apparire la nuova finestra. • <code>LPCTSTR lpWindowName</code> → nome della nuova finestra • <code>DWORD dwStyle</code> → stile della nuova finestra • <code>int x</code> → coordinata x della posizione iniziale della finestra • <code>int y</code> → coordinata y della posizione iniziale della finestra • <code>int nWidth</code> → larghezza della finestra • <code>int nHeight</code> → altezza della finestra • <code>HWND hWindParent</code> → maniglia all'eventuale parent della nuova finestra (il parent fa da contenitore del child) • <code>HMENU hMenu</code> → maniglia a un eventuale menu • <code>HANDLE hInstance</code> → maniglia a un'istanza del processo di riferimento • <code>PVOID lpParam</code> → puntatore a parametri di creazione
OUTPUT	Handle alla nuova finestra in caso di successo, NULL altrimenti

WIN-API	<code>INT GetMessage()</code>
DESCRIZIONE	Manda il thread chiamante in stato di blocco finché non avviene un messaggio-evento.
PARAMETRI	<ul style="list-style-type: none"> • <code>LPMSG lpMsg</code> → puntatore all'area di memoria in cui verrà registrato il messaggio associato all'evento atteso. • <code>HWND hWind</code> → maniglia alla finestra che deve essere colpita dal messaggio-evento che si sta attendendo; se impostata a NULL, vuol dire che tutte le finestre sono di interesse. • <code>UINT wMsgFilterMin</code> → valore minimo dei codici numerici dei messaggi-evento di interesse • <code>UINT wMsgFilterMax</code> → valore massimo dei codici numerici dei messaggi-evento di interesse
OUTPUT	



WIN-API	BOOL TranslateMessage()
DESCRIZIONE	Fa sì che il payload presente in un particolare messaggio, che in taluni casi può essere generato per effetto di attività dell'utente interattivo, sia trasformato affinché rappresenti in codifica corretta i caratteri che eventualmente corrispondono alle chiavi che l'utente ha utilizzato su tastiera/mouse /ecc.
PARAMETRI	• const LPMMSG lpMsg
OUTPUT	

WIN-API	LRESULT DispatchMessage()
DESCRIZIONE	Porta il thread chiamante a eseguire la funzione di gestione dei messaggi-evento relativa alla finestra che è stata colpita.
PARAMETRI	• const LPMMSG lpMsg → messaggio che contiene i parametri da passare all'handler.
OUTPUT	

WIN-API	LRESULT SendMessage()
DESCRIZIONE	Invia un messaggio-evento a una o più finestre, ponendolo in coda alla coda dei messaggi-evento ancora da dispatchare. È una funzione sincrona: il chiamante rimane nello stato di blocco finché il messaggio-evento non sarà stato processato dal thread che effettua il polling. Nel caso specifico in cui il messaggio-evento viene inviato a una finestra propria del chiamante, è proprio quest'ultimo a eseguire subito il dispatching dell'handler.
PARAMETRI	• HWND hWnd → handle verso la finestra da colpire; se impostato a HWND_BROADCAST, il messaggio verrà inviato a tutte le finestre prive di genitore (top level). • UINT Msg → codice numerico del messaggio-evento da inviare. • WPARAM wParam • LPARAM lParam
OUTPUT	



WIN-API	BOOL PostMessage()
DESCRIZIONE	Invia un messaggio - evento a una o più finestre, ponendolo in fondo alla coda dei messaggi - evento ancora da dispatchare.
PARAMETRI	<ul style="list-style-type: none">• HWND hWnd• UINT Msg• WPARAM wParam• LPARAM lParam
OUTPUT	

WIN-API	UINT RegisterWindowMessage()
DESCRIZIONE	Crea un nuovo tipo di messaggio - evento.
PARAMETRI	<ul style="list-style-type: none">• LPCTSTR lpString → nome del nuovo messaggio - evento; se è già esistente, la creazione non avviene e la API restituisce il codice numerico corrispondente.
OUTPUT	Codice numerico del messaggio - evento in caso di successo, 0 altrimenti.



SOCKET

WIN-API	int WSASStartup()
DESCRIZIONE	Initializza l'interfaccia Winsocket per il processo chiamante affinché esso possa utilizzare i socket. • WORD wVersionRequested → versione della Winsocket che si vuole utilizzare per il chiamante.
PARAMETRI	• LPWSADATA lpWSAData → puntatore a una struttura in cui verrà restituita la versione della Winsocket che eventualmen- te verrà attivata.
OUTPUT	0 in caso di successo, un codice di errore altrimenti

Tutte le altre API sui socket sono identiche a quelle di UNIX eccetto per l'utilizzo di handle (identificati col tipo di dato SOCKET) anziché fi descrittore per effettuare operazioni sui relativi oggetti di I/O.